NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN
87-2

EXPERT SYSTEM ENHANCEMEN. TO THE RESOURCE ALLOCATION MODULE OF THE NCS EMERGENCY PREPAREDNESS MANAGEMENT INFORMATION SYSTEM (EPMIS)

JANUARY 1987

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION / AVAILABILITY OF REPORT Distribution Unlimited		
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NCS TIB 87-2			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Delta Information Systems	6b. OFFICE SYMBOL *(If applicable)*		7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS *(City, State, and ZIP Code)* Horsham Business Center, Bldg. 3 300 Welsh Road Horsham, PA 19044			7b. ADDRESS *(City, State, and ZIP Code)*		
8a. NAME OF FUNDING / SPONSORING ORGANIZATION National Communications System	8b. OFFICE SYMBOL *(If applicable)* NCS-TS		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DCA100-86-C-0099		

8c. ADDRESS *(City, State, and ZIP Code)* Office of Technology & Standards Washington, D.C. 20305-2010	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO.	PROJECT NO	TASK NO.	WORK UNIT ACCESSION NO.
			1	

11. TITLE *(Include Security Classification)*
Expert System Enhancement to the Resource Allocation Module of the NCS Emergency Preparedness Management Information System (EPMIS)

12. PERSONAL AUTHOR(S)

13a. TYPE OF REPORT Final	13b. TIME COVERED FROM _____ TO _____	14 DATE OF REPORT *(Year, Month, Day)* January 1987	15 PAGE COUNT 229

16. SUPPLEMENTARY NOTATION

17.	COSATI CODES		18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Emergency Preparedness Management Information System (EPMIS) Resource Allocation Module (XTRAM)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The purpose of the program discussed in this report is to develop an Expert System Enhancement to the Resource Allocation Module (XTRAM) for EPMIS. The project consisting of the five tasks outlined below is underway. Task 1 has been completed, and this report summarizes all the work performed on this effort. Tasks 2 through 5 are options which may be exercised by the NCS following the evaluation of Task 1.

TASK	TITLE
1	XTRAM System Design Specification
2	System Acquisition
3	Developing the XTRAM Knowledge Base
4	User Documentation
5	XTRAM Expert System Demonstration

Work on Task 1 has been divided into the five subtasks which are listed below. The work performed on each of the subtasks is summarized in the indicated sections of this report.

(over)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dennis Bodson	22b. TELEPHONE *(Include Area Code)* 202-692-2124	22c. OFFICE SYMBOL NCS-TS

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

Item #19 Cont'd:

Report Section	Subtask
2.0	Requirements Analysis
3.0	Analysis of Hardware Alternatives
4.0	Analysis of Software Alternatives
5.0	Detailed Design Specification
6.0	EPMIS Compatibility Plan

NCS TECHNICAL INFORMATION BULLETIN 87-2

EXPERT SYSTEM ENHANCEMENT TO THE RESOURCE ALLOCATION MODULES
OF THE NCS EMERGENCY PREPAREDNESS MANAGEMENT INFORMATION SYSTEM (EPMIS)

January 1987

PROJECT OFFICER   APPROVED FOR PUBLICATION:

DENNIS BODSON   DENNIS BODSON
Senior Electronics Engineer   Assistant Manager
Office of NCS Technology   Office of NCS Technology
  and Standards   and Standards

FOREWORD

The National Communications System (NCS) is an organization of the Federal
Government whose membership is comprised of 22 Government entities. Its mission
is to assist the President, National Security Council, Office of Science and
Technology Policy, and Office of Management and Budget in:

   o   The exercise of their wartime and non-wartime emergency functions and
        their planning and oversight responsibilities.

   o   The coordination of the planning for and provision of National
        Security/Emergency Preparedness communications for the Federal
        Government under circumstances including crisis or emergency.

In support of this mission, the NCS has developed the Emergency Preparedness
Management Information System (EPMIS) to permit the Manager, NCS and the
designated Resource Allocation Officer (RAO) to respond effectively to declared
national emergencies. This is in direct support of the survivability and
endurability objectives addressed by Executive Order 12472 and National Security
Decision Directive 97. This report represents a system design specification of
the Expert System Enhancement to the Resource Allocation Module (XTRAM) of
EPMIS. XTRAM will assist the RAO in utilizing EPMIS for allocation and use of
limited telecommunication assets in times of crises and emergencies.

Comments on this TIB are welcome and should be addressed to:

                                   Office of the Manager
                                   National Communications System
                                   ATTN: NCS-TS
                                   Washington, DC 20305-2010
                                   (202)-692-2124

EXPERT SYSTEM

ENHANCEMENT TO THE RESOURCE ALLOCATION

MODULE OF THE NCS EMERGENCY PREPAREDNESS

MANAGEMENT INFORMATION SYSTEM (EPMIS)

January, 1987

Task 1 Final Report

SYSTEM DESIGN SPECIFICATION

Submitted to:

NATIONAL COMMUNICATIONS SYSTEM

Office of Technology and Standards

Washington, DC   20305

Contracting Agency:

DEFENSE COMMUNICATIONS AGENCY

Contract Number - DCA100-86-C-0099

D E L T A   I N F O R M A T I O N   S Y S T E M S,   I N C.

Horsham Business Center, Bldg. 3

300 Welsh Road

Horsham, PA 19044

TABLE OF CONTENTS

1.0 Introduction

This document summarizes work performed by Delta Information Systems, Inc., for the Office of Technology and Standards of the National Communications System, an organization of the U.S. Government, under Contract number DCA100-86-C-0099. The Office of Technology and Standards, headed by National Communications System Assistant Manager Dennis Bodson, is responsible for the development of advanced technology for the NCS.

BACKGROUND

The Office of the Manager, NCS requires timely, accurate information about the status of communications resources during national emergencies and declared disasters. This led to the realization that an automated decision support system would be useful to NCS Emergency Management Teams, which play a significant role in the monitoring and resolution of such situations.

The Emergency Preparedness Management Information System (EPMIS) is an integral part of the National Emergency Telecommunications Management System, which is designed to allow the Manager, National Communications System (NCS) to respond to declared national emergencies. The EPMIS is a user-oriented, decision-support tool designed to assist the Manager, NCS in the performance of his assigned emergency communications management mission by providing timely information about the residual communications capabilities and the outstanding National Security

1 - 1

Emergency Preparedness (NSEP) communications requirements of the
nation.

OVERVIEW

The purpose of the program discussed in this report is to
develop an Expert System Enhancement to the Resource Allocation
Module (XTRAM) for EPMIS. In accordance with the Statement of
Work, a project consisting of the five tasks outlined below is
underway. Task 1 has been completed, and this report summarizes
all the work performed on this effort. Tasks 2 through 5 are
options which may be exercised by the NCS following the
evaluation of Task 1.

TASK	TITLE
1	XTRAM System Design Specification
2	System Acquisition
3	Developing the XTRAM Knowledge Base
4	User Documentation
5	XTRAM Expert System Demonstration

Work on Task 1 has been divided into the five subtasks which
are listed below. The work performed on each of the subtasks is
summarized in the indicated sections of this report.

REPORT SECTION	SUBTASK
2.0	Requirements Analysis
3.0	Analysis of Hardware Alternatives
4.0	Analysis of Software Alternatives
5.0	Detailed Design Specification
6.0	EPMIS Compatibility Plan

RECOMMENDATION

The Detailed Design Specification of the proposed
hardware/software alternatives for the XTRAM system includes one
hardware system, and two knowledge software systems. The
hardware system, as recommended in Section 3, is a DEC VAXstation
II workstation. The final two knowledge software alternatives
to be considered for XTRAM are ART by Inference, and S.1 by
Teknowledge. Of the two alternatives ART is recommended for
implementation.

2.0 Requirements Analysis

The function of XTRAM is to serve as an advisor or consultant to the Resource Allocation Officer (RAO) in allocating scarce communications resources, especially in times of national emergency. Currently, the RAO has the use of the Emergency Preparedness Management Information System (EPMIS), which is basically a custom-designed file management system that keeps track of the availability of communications resources, and the demands upon them.

For the most part, XTRAM will obtain pertinent information from EPMIS, just as the RAO would do without XTRAM, in order to produce recommended resolutions to requests for Telecommunications services. The nature of the information stored in EPMIS has changed somewhat since the XTRAM Statement of Work was issued in 1985. One important change is that there are no longer requests (formerly Claims) for Resources, such as Nodes, Links, mobile terminals, HF sets, etc. Now there are only Service Requests (SR), which require a circuit connecting source and destination users. Each SR is for a single circuit only; requests for multiple circuits between the same users result in multiple SR's.

The priority of each SR is determined by EPMIS, based on the Telecommunications Service Priority (TSP), the function, the Emergency Time Line (ETL), and when the SR was received. A

prioritized list of SR's is presented to the RAO. From this list

the RAO selects an SR to resolve, and may activate XTRAM to

provide recommended resolutions.

The RAO may also indicate other SR's that he wants XTRAM to

consider while resolving the one SR. In this way, subsequent

SR's will have a better chance to be resolved, since limited

Resources that can resolve more than one SR will tend not to be

used up on the first SR.

EPMIS permits the inclusion of data on specific Links within

a Network. At present, there are no plans to incorporate Link

data in the initial data-gathering effort. However, XTRAM will

be capable of utilizing Link data where it exists. For Networks

with Link data, specific paths through the Network will be

evaluated. For Networks without any Link data, XTRAM will assume

connectivity between any two operational Nodes of the Network.

Similarly, EPMIS allows for the inclusion of data on mobile

Assets, such as satellite terminals or HF radios, and the Asset

Centers where they are located. Again, there is no current plan

to include this data in the initial data-gathering effort.

However, XTRAM will attempt to use Assets to resolve SR's, where

data on Assets exists. Where no such data exists, only the fixed

Network facilities will be used to resolve SR's.

The output of XTRAM will be an ordered list of possible

resolutions to a single SR. The list is ordered by XTRAM's

estimate of the desirability of each possible resolution. The

RAO may select any of the listed resolutions, or may ignore

XTRAM's advice and devise his own resolution, which he can then implement using EPMIS.

If the RAO selects one of the XTRAM choices, he can have XTRAM "implement" the resolution by taking the following steps:

1) The resolution description displayed to the RAO will be copied into the appropriate slot (RESOLTN_DESC) in the EPMIS data base. This description is text, limited to 252 characters.

2) The Point of Contact (POC) information will be copied into the appropriate slot (RESOURCE_POC) in EPMIS. The POC will be from the same agency as the source or destination user, if possible, and will likely be from one of the nearby Nodes used in the resolution.

3) XTRAM will indicate, in a field to be added to EPMIS, the number of times that a Link has been used in the resolution of a SR. This information will, in turn, be used in resolving subsequent SR's by avoiding over-use of a particular Link.

4) XTRAM will place `IN USE' in the Availability field of an Asset when the Asset has been used in the resolution of a SR. This information will of course affect the resolution of subsequent SR's.

XTRAM will not print or journal SR's. These are one-stroke operations that can be performed by EPMIS.

In addition to selecting a resolution from the list presented by XTRAM, the RAO may ask several questions of XTRAM in order to be convinced that he is selecting the best possible resolution.

2 - 3

The first question is "Why?". This can be asked about any of the listed resolutions, and XTRAM will respond with the principal reasons why that particular resolution was a strong candidate, expressed in English-like text.

Another question that may be asked by the RAO is "What if?". In this case the RAO may postulate some change in the availability of resources, the requirements of the SR, or the applicability of a particular rule in XTRAM, and XTRAM will redo its recommendations. The changes to be made will be in simple English-like text or through the use of menus.

The last question that may be asked by the RAO is "Why not?". In this case, the RAO assembles a resolution to the SR that he thinks is good, but which was not listed by XTRAM as a possible resolution. The resolution is assembled by the use of menus. XTRAM then evaluates this resolution and gives reasons why it was not selected in English-like text. This feature allows the RAO to select a resolution not recommended by XTRAM, and to test it to determine if there are any flaws in it that he might not have thought of.

In addition to their operational utility, the above question features will be useful during the development of XTRAM.

While almost all of the information XTRAM requires will be obtained from the EPMIS data base, XTRAM may obtain some few pieces of information by directly asking the RAO. This would be done only in those cases where the required information does not reside in EPMIS, where there is a definite answer to the query,

and where theie is at least some expectation that the RAO will know the answer. Most questions will be able to be answered by multiple choice, a numeric value, or a name (such as a Node name). When asked a question, the RAO may decline to answer, or may ask "Why?", in which case XTRAM will present reasons why having this information is important in recommending resolutions.

Rules for XTRAM will be obtained from an expert RAO. However, the following preliminary rules will be used as a starting point:

1) When possible, a fixed Network, rather than mobile Assets will be used.

2) A Network will be used that belongs to the requesting Agency.

3) Nodes close to the source and destination users will be used, in order to minimize the difficulty of local connections.

4) Nodes belonging to the same Network will be used in order to avoid having to transfer from one Network to another.

5) Facilities will be used that have a high likelihood of being operational, based on actual and predicted status.

6) If fixed Network facilities cannot be used, then use mobile or other Assets from Asset Centers that are near the source and destination users.

7) High capacity Assets already deployed to other users nearby may also be used.

8)   Use Links and Assets that are compatible with the
Service Request in terms of data or voice.

9)   Where Links are defined for a Network, use as few as
possible, regardless of route miles.

10)   Downgrade a resolution that utilizes a limited Resource
that is also used in the first-choice resolution of another
Service Request under consideration.

The objective of many of the rules is to satisfy the Service
Request in the most efficient manner, so that subsequent Service
Requests, whether currently in the queue or not yet received, can
be resolved.

3.0 Analysis of Hardware Alternatives

In selecting hardware to satisfy the XTRAM functional requirements, several factors were considered in the decision-process. These include DEC compatibility, availability of software, computing power, ease of database interface, programming facilities, and price.

3.1 Criteria

An important consideration in choosing hardware for application development is the ease with which the application can be moved from development to delivery. Compatibility between hardware systems, or using the same system for development and delivery ensures that the application can be run without modification. This factor was heavily stressed by NCS. Since the EPMIS system resides on a VAX, the XTRAM hardware should be DEC compatible for future XTRAM/EPMIS integration. The more DEC compatible the prototype, the easier an XTRAM integration with EPMIS will be.

As in traditional computing, software availability is an important consideration in the XTRAM hardware selection. It will be desirable to utilize knowledge-engineering software together with traditional languages and software packages. It is important to choose hardware which will run expert systems both

powerful and flexible enough to satisfy the XTRAM functional
requirements.

Another important factor considered was computing power.
Knowledge based programs frequently require a larger address
space and more machine cycles than traditional programs. A large
address space is an advantage because it allows the program to
address many locations without requiring the hardware or software
to do a great deal of context switching, which would slow down
processing. Also, knowledge system programs tend to tackle
complex problems that require operations such as searching or
pattern matching. For this reason, speed is likely to be more of
a consideration in a computer used for knowledge systems work
than for one used in regular data processing. In addition, some
"number crunching" capability is required to perform the spacial
and temporal computations that are part of resource allocation.

The ease with which the XTRAM computer could be connected
to, and integrated with the EPMIS computer was also a factor.
Since EPMIS resides on a DEC MicroVAX, it is desirable that the
hardware considered is able to support DECnet network software,
and an Ethernet port for fast, efficient data communications.
Also, since the EPMIS database is managed by the INGRES database
management system, the hardware considered must be able to
support INGRES networking and user interface software to ensure
an efficient database access.

Finally, as in most systems, good programming facilities are important because the rate at which program development progress is made is strongly influenced by the ease with which developers interact with the computer.

3.2 The Alternatives

The four leading hardware candidates DIS evaluated were the IBM PC/AT, the Symbolics 3620 LISP Machine, the Sun 3/100 160 workstation, and the DEC VAXstation II. The IBM PC/AT is widely used for many applications, most of which are business as well as personal applications. The last three, the LISP Machine, the Sun 160, and the VAXstation, can be considered workstations. Although workstations are generally thought of as research machines, some of them on the less expensive end of the scale are appropriate as delivery vehicles.

3.2.1 IBM PC/AT

An IBM PC/AT may be thought of as a low-end workstation; that is a PC may be able to perform all of the functions of a workstation, but at reduced speed, with lower resolution graphics, less memory, etc.

The PC has limited computing power with a maximum of 1.3M of memory. This limits the AI software availability for the PC.

For the most part, the expert system shells available for the PC
are not flexible or powerful enough to solve significant
problems.

As for compatibility, the PC does not support DECnet
software, nor does it have an Ethernet port. If a physical
connection between the PC and the VAX were to be made, it would
have to be over an RS-232 communication link which is on the
order of 250 times slower than an Ethernet communication link.
Currently, there is no INGRES network software or user interface
software available for a PC. This would make it very difficult
to communicate efficiently with the EPMIS database residing in
the EPMIS VAX. Also, the expert system software as well as the
database interface software could not be easily ported to a VAX
for any future EPMIS/XTRAM integration.

3.2.2 Symbolics 3620 LISP Machine

The Symbolics LISP machine is an AI workstation. It is a
very powerful symbolic processing machine. The hardware
architecture is built to process LISP code in which most expert
systems are implemented. It provides a multitasking,
multiwindowing environment with excellent program development
facilities as well as high resolution graphics.

3 - 4

AI software availability is not a problem for the LISP
Machine. It can run very powerful and flexible expert systems
quickly and efficiently. Some of the expert systems which can
run on the LISP machine have versions which can also run on a VAX
system. There may be differences, however, between the LISP
Machine and VAX versions of the expert systems depending on the
specific expert system considered. For this reason, there may be
some work* involved in making the expert system software VAX
compatible. The LISP Machine is equipped with an Ethernet port
and supports DECnet software which would make a physical
connection to the EPMIS VAX possible.

The problem however, with the LISP Machine is that it does
not support any INGRES software. This would make it extremely
difficult if not impossible to communicate with the EPMIS
database efficiently. Since a good deal of the processing to be
done involves the database interface, this is a very significant
problem.

Some of the features of this machine are; up to 16M bytes of
memory, 360M bytes of disk space, and a high resolution
monochrome graphics monitor.

---

* The work required could be significant depending on the
VAX version of the expert system software

3.2.3 Sun 3/100 160

The Sun 3/100 160 is considered a general purpose
workstation. Using the numeric Motorola 68020 processor, the Sun
provides a general purpose computational engine with enough power
for AI applications. It is a UNIX based system with high
resolution graphics, multiwindowing and multitasking
capabilities.

Sun has its own version of LISP, referred to as Sun Common
LISP. Sun Common LISP adheres completely to the Common LISP
standard proposed by the Defense Advanced Research Projects
Agency and is fully integrated with the UNIX operating system.
Because of Sun's compatibility with Common LISP, AI software
availability on the Sun would not be a problem. The Sun can run
many of the powerful and flexible LISP implemented expert system
shells which run on the LISP Machine. With Sun's processing
power, the LISP processing speed on a Sun is comparable to that
of a LISP Machine. In the latest AI software trend, many AI
software companies are making "C" versions of their expert system
shells. These "C" versions may run even faster on the Sun, than
LISP versions of the same product.

The Sun is a general purpose, numeric machine. Many
traditional languages and software packages are available on the
Sun including INGRES network and User Interface software which is

an important feature in the XTRAM system. With INGRES software

and networking available, a fast, efficient access to the EPMIS

database access could be achieved. In addition, the Sun is

equipped with an Ethernet port for fast data transfers.

Currently, the Sun communication protocol is not DECnet

compatible, but will be by early 1987.

As for software portability to a VAX system, the same

applies for expert system software portability for the Sun as it

did for the LISP Machine. There may be differences between the

Sun and VAX versions of the expert systems depending on the

specific expert system considered. However, if "C" versions of

the expert systems are considered, there should be fewer

differences between the Sun and VAX versions of the expert system

software as compared to the LISP Machine and VAX versions because

of the excellent portability characteristics of the "C" language.

Also, since the same traditional software languages which run on

a Sun are available on a VAX, the software interface source code,

with minor alterations, could also be made to run on a VAX.

Some of the features of the Sun are: up to 16M bytes of

memory, up to 1.1G bytes of disk space, a high resolution

monochrome monitor and an Ethernet port.

3.2.4 DEC VAXstation II

The DEC VAXstation II can be considered a general purpose workstation. It is a very powerful machine and provides a good program development environment. It also provides multitasking, multiwindowing, and high resolution graphics.

Unlike the LISP Machine, which devotes its processing power to LISP processing, the VAX's processing power is devoted to more traditional computing tasks. This would be advantage for the VAX over the LISP Machine in performing the expert system interface processing involved in communicating with the EPMIS database. The VAX has a disadvantage in LISP processing power as compared to the LISP Machine, however, many expert system shells are now being written in "C" which the VAX will can execute considerably faster than it can execute LISP.

Software availability for the VAXstation would not be problem. It can run many of the same expert system shells which run on the LISP Machine. Also, since the VAXstation is in the VAX family, it supports all INGRES software as well as many of the traditional languages such as "C" and FORTRAN which may be used in the database interface.

Since a VAXstation II is a single user Microvax, and the EPMIS system will be implemented on a Microvax, hardware and

software compatibility would not be a problem. Because of this
compatibility, the expert system and integrated software on a
VAXstation can be easily ported to another VAX with little or no
work. Some of the VAXstation features are; up to 9M bytes of
memory, 159M bytes of disk space, a high resolution monochrome
monitor, and an Ethernet port.

3.3 Summary

Figure 3-1 shows a general comparison of the four
alternatives mentioned above. Figure 3-2 is a chart showing the
computing power of the four alternatives after the hardware
analysis performed by DIS based on the criteria mentioned, and
the desires of NCS, it would seem that the DEC VAXstation II is
the best candidate. Using a DEC VAXstation II with its excellent
DEC and INGRES compatibility along with its proce_sing power,
would satisfy the XTRAM functional requirements as well as
provide a step towards EPMIS/XTRAM integration.

XTRAM HARDWARE ALTERNATIVES COMPARISON CHART

○ – POOR
◉ – FAIR
◍ – GOOD
● – EXCELLENT

	DEC COMPATIBILITY	COMPUTING POWER	EXPERT SYSTEMS	EASE OF INTEGRATION	PRICE
IBM PC/AT	○ (POOR)	○ (POOR)	◉ (FAIR)	○ (POOR)	$9,000
SYMBOLICS LISP MACHINE	◉ (FAIR)	● (EXCELLENT)	● (EXCELLENT)	◉ (FAIR)	$50,000
SUN WORKSTATION	◉ (FAIR)	● (EXCELLENT)	● (EXCELLENT)	◍ (GOOD)	$50,000
DEC VAXstation II	● (EXCELLENT)	● (EXCELLENT)	◍ (GOOD)	● (EXCELLENT)	$47,000

Figure 3-1

COMPUTING POWER CHART

Spec System	Processor Type	Processor Size	Max. RAM
IBM PC/AT	Intel 80286	16 bit	3 MB
Symbolics 3620 LISP Machine	Symbolics 3620	36 bit	16 MB
Sun 3/100 160 Workstation	Motorola 68020	32 bit	16 MB
DEC VAXstation II	MICROVAX 78032	32 bit	9 MB

FIGURE 3-2

4.0 Analysis of Software Alternatives

The analysis of the software alternatives was broken into three tasks. First, a set of criteria was developed to perform the analysis. Second, each alternative was analyzed to determine its candidacy for the XTRAM-expert-system role. Lastly, if more than two candidate alternatives remained, the best two were chosen as the XTRAM expert system alternatives.

4.1 Criteria

The criteria developed to analyze and compare the commercial expert system tools are based on contractual requirements and NCS's long term goals for XTRAM. Shown in order of importance the criteria are:

1. The expert system can easily interface with external processes.

2. The expert system can handle uncertain information.

3. The expert system can reside on a DEC Micro VAX computer.

4. The expert system's reasoning process can be controlled by a knowledge engineer.

5. The expert system's knowledge is stored in an organized fashion.

6. The expert system can justify its solutions.

7. The expert system can find a solution in a reasonable amount of time.

8. The development of the expert system is not a
    tedious or time consuming task.

These criteria stress the features desired in the XTRAM expert

system and are discussed in the following sections.

The expert system must be able to interface with external

processes.

For the XTRAM expert system to perform its mission it must have

access to information stored in the INGRES relational data base

system. This information can be accessed in one of two ways. The

first is through an INGRES interface built into the expert system,

and the second is through an interface built by the knowledge

engineer. In the former case the interface, supplied by the vendor,

is tightly coupled to the expert system and the knowledge engineer

does not have to do any additional programming. In the latter case

the interface is loosely coupled and must be designed, built, and

implemented by the knowledge engineer. In addition, the interface

is invoked, by the expert system, via external function calls.

The built-in interface is preferred because the knowledge

engineer does not have to build the interface and the development

effort is reduced. However, an expert system capable of calling

external functions provides the knowledge engineer with additional

4 - 2

capability besides interfacing with INGRES. The knowledge engineer can connect functions which perform tasks more efficiently than the expert system, for example: calculating the distance between two nodes. Of course the best expert system would exhibit both characteristics, a built-in INGRES interface and an ability to invoke external functions. Thus the knowledge engineer would not have to build an INGRES interface and would be able to add enhancements.

The expert system should be able to handle uncertain information.

XTRAM has to work with uncertain knowledge, such as the certainty of the operational status of a resource. This can be accomplished in two ways. One is if the uncertainty-handling feature is built into the expert system, and the second is if the feature can be added by a knowledge engineer. The former is convenient because the knowledge engineer does not have to build the feature, but control of the certainty-handling algorithm is lost. In the latter, the knowledge engineer programs the feature into the expert system and has control of the algorithm. Unfortunately this increases the development effort.

Either method is acceptable and our ideal expert system would incorporate both. A certainty algorithm would be built-in and the knowledge engineer would have the option to modify it.

4 - 3

A DEC Micro VAX computer is the preferred host for the expert system.

XTRAM is part of EPMIS, which resides on a Micro VAX computer, and the integration of both would be simplified if XTRAM also resided on the same computer. Unfortunately many of the commercial expert system tools are only available on non-VAX computers. Some of these tools do, however, permit the developed expert system to be delivered on a Micro VAX. They accomplish this by supplying the complete tool on a non-VAX computer and when development is completed a stripped down delivery version permits the expert system to run on a Micro VAX. The delivery version does not permit the user to change the expert system; only the development version can do that. Also, only the central location would have a non-VAX computer; all the users would have a Micro VAX. This approach assumes changes to the expert system are to be performed at a central location and all users are to have identical copies. If the users need to change the expert system, then a development version is needed by each user along with a non-VAX computer, or else only a tool available on the Micro VAX can be used.

However, compliance with the Micro VAX should not rule out non-VAX tools lacking a delivery Micro VAX vehicle. The tool might be XTRAM's best candidate and the vendor's product plan might include eventual migration to the VAX. So this criterion was expanded to consider a non-VAX tool if it has a delivery Micro VAX

4 - 4

vehicle or if the vendor's product plan includes eventual migration to the Micro VAX. The ideal tool, however, would meet other all criteria and reside on a Micro VAX computer.

The knowledge engineer should be able to control the expert system's reasoning process.

XTRAM is in an evolutionary stage and to accommodate new demands the embedded expert system should have a flexible reasoning process. Expert systems use two types of reasoning processes. The first is forward chaining and is best suited to problems with a large number of solutions, such as the number of ways to link two nodes in a network. The second method, backward chaining, is suited to diagnostic problems where the goal is known and there are few solutions; for example, determining if a node is a connective point for two networks. Both methods could be used in XTRAM and our ideal expert system would permit the knowledge engineer to invoke either one alone or in combination.

The expert system's knowledge should be stored in an organized fashion.

XTRAM handles interrelated and hierarchal knowledge. For instance, a government agency may own two networks and one of the networks might lease lines from a commercial network. (See Figure 4-1.) To handle this knowledge XTRAM needs a corresponding
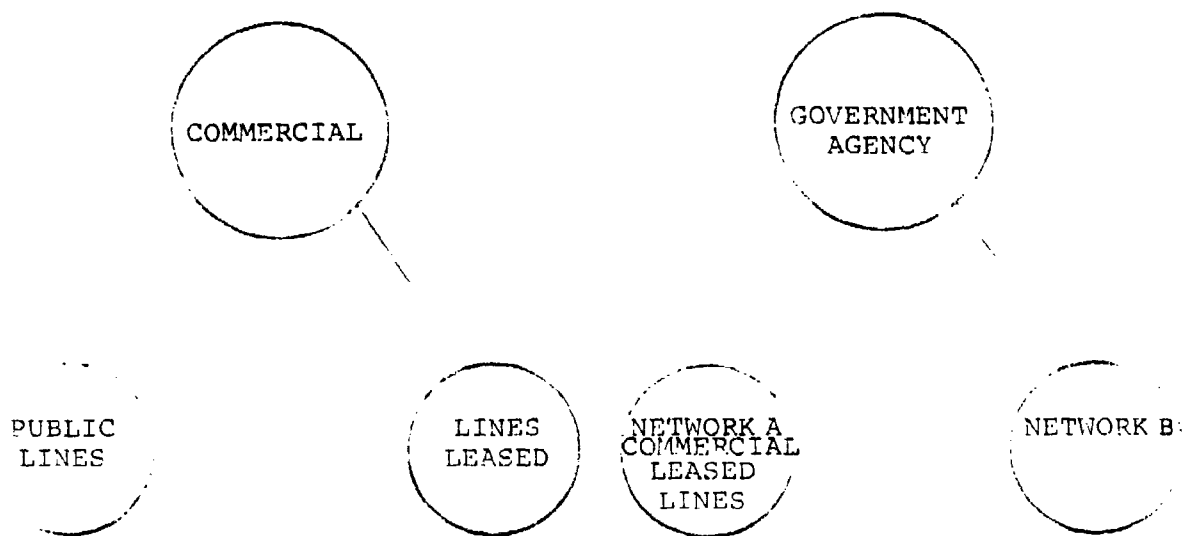
4 - 5

FIGURE 4-1: A possible Government Agency Network

capability in the expert system. Three methods are currently used by expert systems to store knowledge: attribute-value (A-V) pairs, object-attribute-value (O-A-V) triplets, and frames. An A-V pair is found in simple expert systems which are built around one undeclared object. For example, an expert system might exist for determining the best paint color to use to cover over a car's original color. The car would be the assumed object and the A-V pair would be original-color (attribute) and the color red (value). (See Figure 4-2.)

An O-A-V triplet has the object explicitly declared which permits multiple objects, and permits the objects to be interrelated. For instance, using our previous example we have a car (object) whose original-color (attribute) is the color red (value) and whose owner-is (attribute) Smith (value) and Smith's (object) body-height (attribute) is 5'11" (value). (See Figure 4-3.)

A frame is a description of an object which contains slots for all information related to the slot. For instance, if there is a frame called "car" it would have slots "color" and "owner". (See Figure 4-4.) These slots would contain the values "red" and "Smith" respectively. The slots permit a richer representation of knowledge and may contain default values, pointers to other frames, sets of rules or procedures to obtain values.
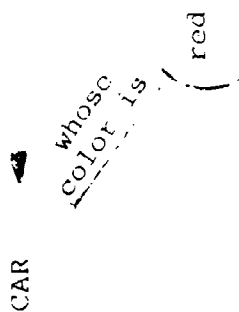
Assumed
Object

CAR → whose color is (red)

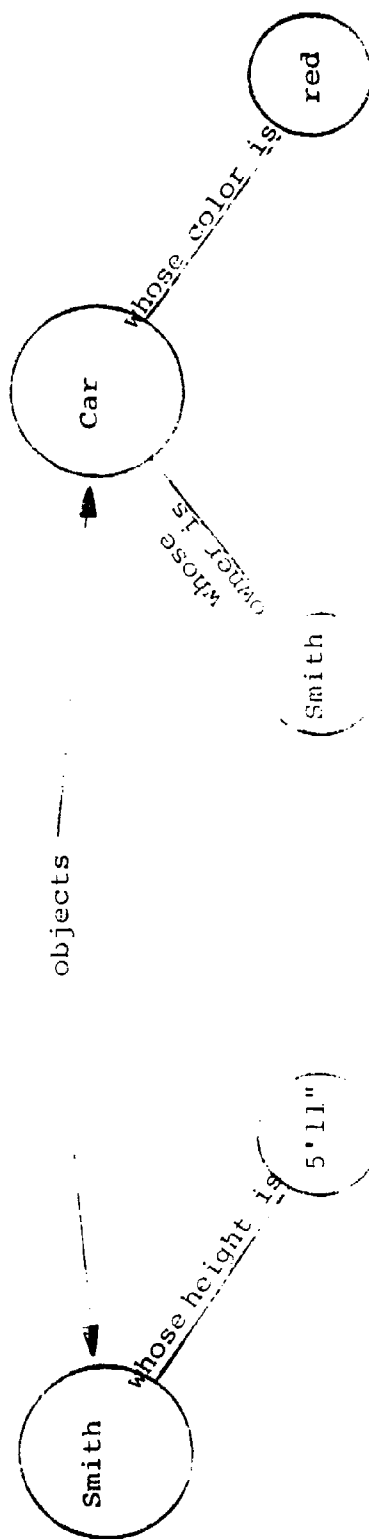FIGURE 4-2: Attribute-Value Pair

objects ———

Car → whose color is → (red)

whose owner is (Smith)

Smith → whose height is (5'11")

FIGURE 4-3: Object-Attribute-Value Triplets

4 - 8

```
                    ┌─────────────────────────────┐
                    │            C A R            │
                    ├──┬──────────────────────────┤
                    │S │    OWNER          SMITH   │
                    │  │    ┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄┄  │
                    │L │    ORIGINAL               │
                    │  │                           │
                    │O │                    RED    │
                    │  │    COLOR                  │
                    │T │    ───────────────────────│
                    │  │       •                   │
                    │S │       •                   │
                    │  │       •                   │
                    └──┴──────────────────────────┘
```

FIGURE 4-4:   Description of an Object
                    Using a Frame

The last two methods, O-A-V triplets and frames, permit the storage of interrelated and hierarchal knowledge; the form XTRAM's knowledge has. Thus our ideal expert should store knowledge using either O-A-V triplets or frames.

The expert system should be able to justify its solutions.

A user may want to question a solution supplied by XTRAM, such as why a particular solution was chosen or what makes another solution unacceptable or how a solution would be affected if the knowledge base were different. These situations are, in essence, three questions: why, why not and what if. 'Why' forces the expert system to explain the reasons for choosing the given solution. 'Why not' permits the user to have the expert system explain why it didn't choose an alternate solution and 'what if' allows the user to create hypothetical situations and observe the effect on the solution. The ideal expert system would supply all three or have provisions to add them.

The expert system should find a solution in a reasonable amount of time.

Because XTRAM is supposed to provide solutions in normal and crisis situations, it should provide an answer in a reasonable amount of time. During a crisis the solution is needed as soon as possible before the situation becomes worse or snowballs into a

catastrophe. Therefore our ideal expert system should provide an answer in a reasonable amount of time.

The development of the expert system should not be tedious or time consuming.

The development time allocated for the prototype XTRAM is less than six months. Typically if an expert system were to be built from scratch, without using available tools, the development time could take from one to two years. This amount of time is not available for XTRAM, thus commercially available tools are to be used which support the development time. In general, the tool which removes the knowledge engineer furthest from the underlying language on which the tool is built, the better. This indicates the tool provides a rich environment for developing expert systems without requiring the knowledge engineer to perform a lot of 'lower-level' coding. Some flexibility is lost because the knowledge engineer is not working with the underlying language, but this is overshadowed by the faster prototyping of the expert system.

Friendliness of the tool goes hand in hand with reducing the prototyping time. If the tool is difficult to use the knowledge engineer may spend an excessive amount of time developing the expert system, but if the tool is friendly and flexible the development time is reduced.

Thus our ideal expert system tool should remove the knowledge engineer as much as possible from the underlying language and should be easy to use.

Summary

Of these eight criteria, interfacing with external processes and handling uncertainties are the most important. Without them the goals of XTRAM may be difficult to achieve. The remaining criteria are less important but certainly desirable. If an expert system tool supports most of these criteria then the tool could still be considered usable for XTRAM.

4.2 The Alternatives

DIS evaluated small to large expert system tools designed, respectively, for personal (IBM) through symbolic computers. (Detailed information on each tool is available in Appendix A.) Expert systems which obviously did not meet XTRAM's criteria were excluded from this analysis. The candidates for XTRAM are:

```
ART      .....................  by Inference
IKE      .....................  by LMI
KEE      .....................  by Intellicorp
KES      .....................  by Software A&E
Knowledgecraft  ..........  by Carnegie Group
M.1      .....................  by Teknowledge
Personal Consultant Plus..  by Texas Instruments
S.1      .....................  by Teknowledge
```

The criteria discussed in section 4.1 was used to evaluate each tool with the data obtained from literature, watching demonstrations, and talking to vendors. (A standard problem was not executed on any system due to time limitations.) Once evaluated, the system tools were compared and the best two were selected as the choices for XTRAM.

4.2.1 ART by Inference

ART, a general purpose tool, is not geared to any specific problem, and has the following features:

o Ability to interface with "external functions"
o Ability to hypothesize
o Resides on a Micro VAX
o Reasons with forward and backward chaining
o Knowledge is stored in frames
o Finds solutions in a reasonable amount of time
o Rules are compiled
o Is not difficult to use

INGRES Interface

ART does not have an INGRES interface, but one can be added by the knowledge engineer, via "external functions".

Uncertainty

ART does not have uncertainty-handling built-in; but the knowledge engineer can implement it via "external functions" or through a set of rules.

Host Computer

ART is available on both symbolic and numeric computers and operates on: the TI Explorer, LMI, DEC VAX and Micro VAX, or Sun computers.

Reasoning Process

ART is a forward chaining tool with backward chaining capability; both may be controlled by the knowledge engineer.

Knowledge

ART stores knowledge in frames which represent objects or classes of objects, its associated attributes, and its memberships in other classes. The frames can be organized into hierarchies in which knowledge about an object can be automatically deduced based on the class to which it belongs. The logical consistency of data is maintained by ART. If, during the execution of ART, an object is disassociated from some class, ART automatically removes the properties that had been deduced for that object based on its membership in the class.

Justification

ART does not have any built-in justification; but it can be implemented through "external functions" and rules. Also, the effort to develop 'what if' and 'why not' justification can be reduced by ART's ability to reason about hypothetical situations without changing the original facts or deductions.

Speed

ART is considered to be one of the fastest expert systems. The speed is achieved by compiling the rules and knowledge base instead of interpreting them, as most expert systems do. In addition, other expert systems allocate memory until it is used up; they do not deallocate the memory when they are done with it. Once the memory is completely used up, they then suspend the inference mechanism temporarily, and deallocate the no longer needed memory. This process is called "garbage collection". Unfortunately, this memory management scheme causes the expert system to arbitrarily shut down in mid-application until "garbage collection" is completed. ART uses a dynamic management method which deallocates memory when it is no longer needed, yielding an apparent increase in speed and reliability.

ART was originally written in LISP and ran on symbolic computers designed specifically to execute LISP programs. When ART was migrated to numeric computers, such as the Micro VAX, a LISP interpreter was needed which added another layer of software and a

4 - 15

corresponding loss in speed. (All expert systems using this approach lose speed.) The current Micro VAX version of ART is considered, by its vendor, to be extremely slow. To gain comparable speeds on both numeric and symbolic computers, ART has been re-written, via a translator program, into "C". The "C" version is currently undergoing beta testing and is scheduled for release in January 1987.

Friendliness

Developing an expert system with ART would not be difficult. The knowledge engineer has to work with the underlying language, LISP or "C", only if procedural functions are needed or to interface with INGRES. The knowledge engineer interface, ART Studio, is acceptable, permitting browsing of the knowledge base, examining facts and rules, etc. In addition, the manuals were extremely well written; the best of all systems examined.

Conclusions

ART should be considered as one of the two possible expert systems for XTRAM. The tool has all the features desired, or they can be programmed. The only drawbacks are: no existing INGRES interface, certainty-handling or justification. All three can be programmed and for the latter two this may actually be an advantage. The knowledge engineer would have complete control of the

4 - 16

certainty-handling algorithm implemented, and would be able to
implement a justification scheme tailored to XTRAM's requirements.
In addition, ART is one of the fastest expert systems and doesn't
perform any "garbage collection".

4.2.2 IKE by LISP Machines Incorporated (LMI)

IKE is an analytically-oriented expert system-building tool that
is best suited for deductive problems, such as diagnostics. IKE is
not suited for problems where the task is to synthesize evidence to
generate solutions. IKE has the following features:

    o Ability to interface with "external functions"
    o Certainty factors
    o Reasons with backward chaining
    o Knowledge is stored in frames

INGRES Interface

IKE does not have an INGRES interface; but one can be added via
"external functions".

Uncertainty

IKE uses certainty factors which range from definitely-false to
unknown to definitely-true. It is a MYCIN-style implementation.
When certainty factors are 'ORed' together, the maximum certainty
factor is used. When certainty factors are 'ANDed' together, the

minimum certainty factor is used. When two certainty factors are combined, one of three rules is used:

Both certainty factors are positive:

$$cf = x + y(1 - x) \; ; \; \text{where } (1 - x) \text{ is the remaining disbelief}$$
or   (See figure 4-5.)
$$cf = x + y - yx$$

One certainty factor is positive and the other negative:

$$cf = (x + y) / (1 - \min(|x|,|y|)) \; ; \; \text{where } (1 - \min(|x|,|y|))$$
ensures commutativity
for more than two pieces
of evidence

Both certainty factors are negative:

$$cf = -|x| + |y|(x - 1) \; ; \; \text{where } (x - 1) \text{ is the remaining}$$
or   belief
$$cf = x + y + yx$$

Host Computer

IKE is available on symbolic computers: LMI Lambda and the TI Explorer. It is not available on numeric computers, including the DEC Micro VAX.

Reasoning Process

IKE is a backward chaining tool with extremely limited forward chaining capability. Neither is controllable by the knowledge engineer.

4 - 18

+1.0

.76

+.5

0

First Conclusion (.6)

Second Conclusion (.4)
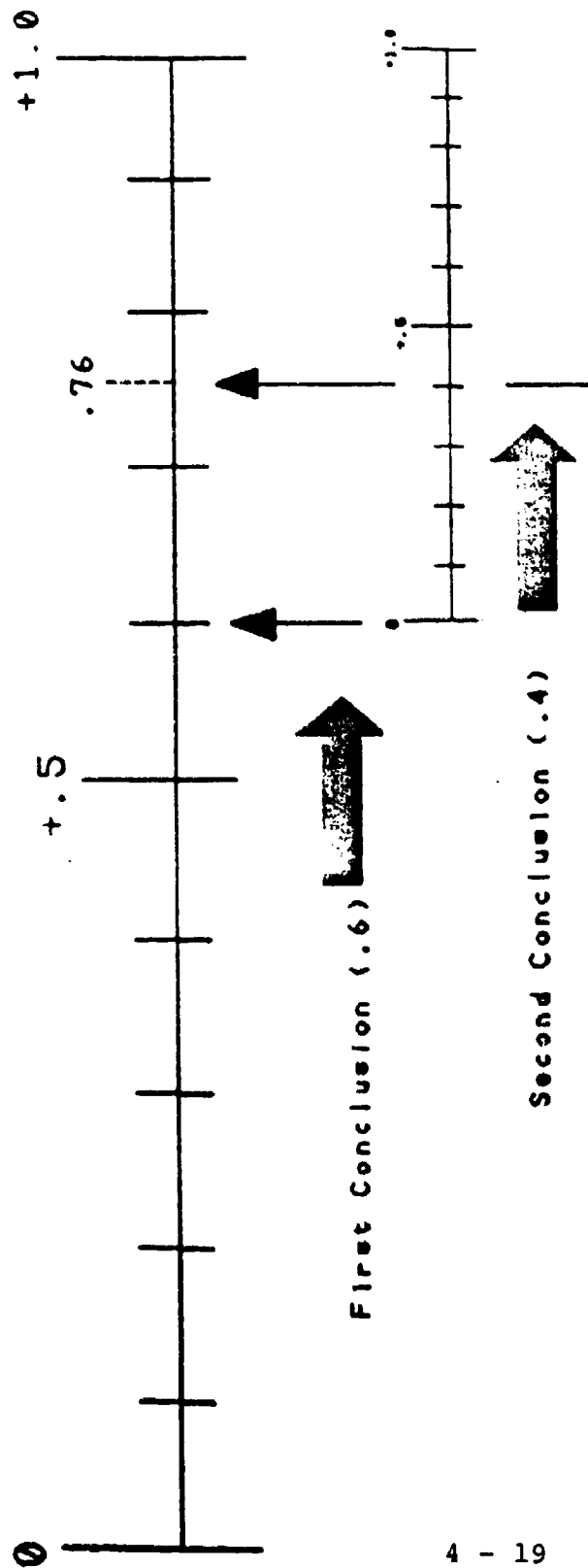
(Resulting certainty is .76)

4 – 19

Figure 4-5: Combining Two Positive Certainty Factors

Knowledge

IKE stores knowledge in frames and permits multiple objects and inheritance. The objects can be constructed into an inheritance tree which establishes a parent-child hierarchy of objects for the purpose of passing attributes from one object to another.

Justification

IKE does not have any built-in justification, but it could be implemented via rules.

Speed

IKE is written in LISP and must stop mid-application to perform "garbage collection".

Friendliness

Developing a goal-directed expert system with IKE would not be difficult. The knowledge engineer interface permits browsing of the knowledge base, examining facts and rules, etc.

Conclusions

IKE should not be considered for XTRAM because of the following major deficiencies: IKE does not currently have a vehicle to migrate to the Micro VAX and the knowledge engineer cannot control the reasoning process.

4.2.3 KEE by Intellicorp

KEE is a general purpose tool, not geared to any specific problem. KEE as the following features:

o Ability to interface with "external functions"
o Ability to hypothesize
o Reasons with forward and backward chaining
o Knowledge is stored in frames
o Finds solutions in a reasonable amount of time
o Is not difficult to use

INGRES Interface

KEE permits the knowledge engineer to add procedures via "external functions".

Uncertainty

KEE does not have uncertainty-handling built-in, but the knowledge engineer can add it via "external functions" or rules.

Host Computer

KEE is available on symbolic and numeric computers. Systems include: Symbolics, TI Explorer, LMI, Xerox 1100, Sun-3, IBM-RT, and the HP9000.

Reasoning Process

KEE can be used in a forward or backward chaining manner, both controllable by the knowledge engineer. In addition, the search strategy in backward chaining, and the conflict resolution in forward chaining, are controllable. (The KEE 3.0 release has the ability to hypothesize.)

Knowledge

KEE stores knowledge in frames, and permits multiple objects and inheritance.

Justification

KEE does not have any built-in justification, but it can be implemented through "external functions" and rules. Also, 'what if' and 'why not' justification development can be reduced by KEE's ability to reason about hypothetical situations without changing the original facts or deductions.

Speed

KEE is written in LISP and must stop mid-application to perform "garbage collection". The hiatus in processing may last several minutes on the Micro VAX; on a symbolic computer it lasts for seconds. The difference in "garbage collection" times is a result of the different hardware employed. A LISP or symbolic computer can execute LISP more efficiently than a numeric computer with a LISP interpreter.

Friendliness

Developing an expert system with KEE would be easy. The rules and data can be hierarchically organized to permit access by multiple knowledge engineers. KEE had the best knowledge engineer interface.

Conclusions

KEE should not be considered for the XTRAM expert system role. Drawbacks are: KEE must perform "garbage collection", and has no certainty-handling or justification, but both can be programmed. In addition, KEE is not currently available on a DEC Micro VAX.

4 - 23

4.2.4 KES by Software A&E

KES is a diagnostic-oriented expert system tool. KES has the following features:

- o Ability to interface with "external functions"
- o Handles uncertainties
- o Reasons with backward chaining
- o Knowledge is stored in A-V pairs

INGRES Interface

KES does not have an INGRES interface, but KES can be added to a "C" program which does, see Figure 4-6. KES would not invoke the interface; instead the "C" program, which calls both the INGRES interface and KES, would be used. The "C" program would provide all interfacing with the outside world, manage the knowledge base, and enable the inference engine. This approach is possible because KES sub-parts were designed to be separated and to work separately.

Uncertainty

KES has several methods available to handle uncertainties. They are: certainty factors, ranking by relative quality, and statistically. Certainty factors handle definitely-false to unknown to definitely-true situations, MYCIN-style. Ranking by relative quality handles poor-average-excellent situations, and statistical
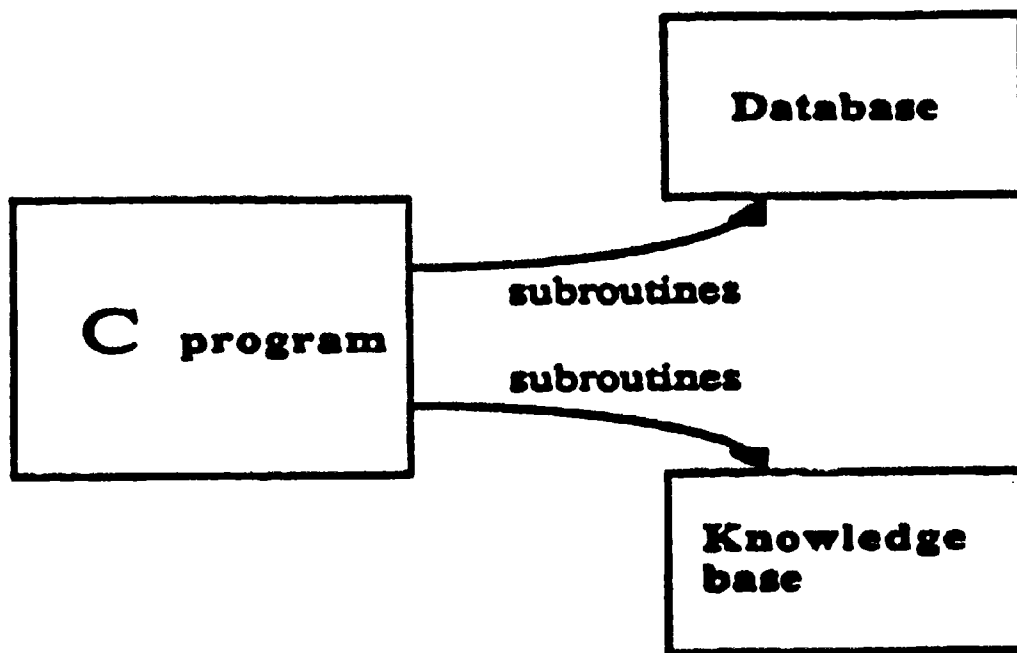
FIGURE 4-6

With KES embedded in the system,
the 'C' program calls both the INGRES
interface and KES.

4 - 25

handles situations involving probabilities.

Host Computer

KES is available on both symbolic and numeric computers, and operates on: the IBM PC, Apollo, Sun, Tektronix and DEC Micro VAX computers.

Reasoning Process

KES is a backward chaining tool. The knowledge engineer has no control over the reasoning process, but can choose one of three reasoning methods. They are: production rules, hypothesize and test, and statistical reasoning. The differences between the methods are in the way knowledge is represented and information is processed. The production rule method is suited to applications where the knowledge is in the form of if-then rules. The hypothesize-and-test method is suited to diagnostic and classification problems. It determines the smallest number of causes that explain the known manifestations of the problem. Lastly, the statistical method performs statistical pattern classification based on Bayes' theorem. It is useful in situations where the pre-existing data is expressed as probabilities.

Knowledge

KES stores knowledge in A-V pairs. The attributes may form hierarchies and be grouped into "classes" to describe an object.

Justification

KES has limited justification capability designed for a dialogue exchange between the user and KES, but it does not fulfill the level of justification desired in XTRAM. However, the capability desired can be programmed into the expert system. The user can query the expert system as to why a question is being asked: 'explain'. The 'explain' probe causes tests associated with the current attribute being requested of the user to be displayed.

Speed

KES is written in "C" and would find a solution in a reasonable amount of time on a numeric computer.

Friendliness

Developing a goal-directed expert system with KES would not be difficult; although KES embedded in a "C" program could get involved.

Conclusions

KES should not be used for XTRAM because the knowledge engineer can not control the reasoning process and the ability to store knowledge is limited.

4.2.5 Knowledgecraft by Carnegie Group

Knowledgecraft is a general purpose tool, not geared to any specific problem, and has the following features:

o Ability to interface with "external functions"
o Ability to hypothesize
o Reasons with forward and backward chaining
o Knowledge is stored in frames

INGRES Interface

Knowledgecraft does not have an INGRES interface, but one can be added, by the knowledge engineer, via "external functions".

Uncertainty

Knowledgecraft does not have uncertainty-handling built-in, but the knowledge engineer can implement it via "external functions" or through a set of rules.

Host Computer

Knowledgecraft is available on both symbolic and numeric computers. Systems include: Symbolics, TI Explorer, DEC VAX and Micro VAX.

Reasoning Process

Knowledgecraft can be used in a forward or backward chaining manner, both controllable by the knowledge engineer. OPS5 provides the forward chaining mechanism and Prolog provides the backward chaining mechanism.

Knowledge

Knowledgecraft stores knowledge in frames, and permits multiple objects and frames.

Justification

Knowledgecraft does not have any built-in justification, but it can be implemented through "external functions" and rules.

Also, the effort to develop 'what if' and 'why not' justification can be reduced by Knowledgecraft's ability to reason about hypothetical situations.

Speed

Knowledgecraft is written in LISP and runs on both symbolic and numeric computers. Knowledgecraft needs a LISP interpreter to operate on a numeric computer and will therefore run slower than a symbolic computer version.

Friendliness

Developing an expert system with Knowledgecraft would be an arduous task. The knowledge engineer interface is poor, and the engineer must work with OPS5 and Prolog to invoke the reasoning process. The rules must be written in an "OPS-like" language which has been described by most users as an extremely difficult language to use. Secondly, the manuals are voluminous and difficult to read.

Conclusions

Although Knowledgecraft is an extremely flexible and powerful tool, and has most of the desired features, it should not be used for XTRAM. Knowledgecraft is too flexible; the knowledge

engineer would expend an excessive amount of time preparing the XTRAM prototype. In addition, the knowledge engineer would be slowed by the poor knowledge-engineer interface.

4.2.6 M.1 by Teknowledge

M.1 can be used to build goal-oriented expert systems. M.1 has the following features:

- o Ability to interface with "external functions"
- o Reasons primarily with backward chaining
- o Knowledge is stored in A-V pairs
- o Is not difficult to use

INGRES Interface

M.1 does not have a INGRES interface, but one can be added via "external functions".

Uncertainty

M.1 uses certainty factors which range from definitely-false to unknown to definitely-true. The associated numerical range is from -100 to 100 ($-100 <= X <= 100$; X is a certainty factor), with unknowns ranging from -20 to 20, ($-20 <= X <= 20$; X is a certainty factor).

Certainty factors are combined such that the final certainty factor is independent of the order in which evidence is found. As positive evidence accumulates the resulting certainty factor approaches but cannot pass 100. Once the certainty of a conclusion reaches 100 it cannot be changed. Equal positive and negative evidence will exactly cancel (except +100 and -100 which cannot be changed once concluded). These rules are modeled after the certainty factor rules of the MYCIN expert system.

Host Computer

M.1 is available on numeric computers such as: the IBM PC, XT and AT.

Reasoning Process

M.1 is a backward chaining tool with extremely limited forward chaining capability. The forward chaining capability permits functions to be invoked and values for attributes to be sought. The inference engine is almost always trying to chain 'backwards' so the knowledge engineer has extremely limited control of the reasoning process.

Knowledge

M.1 stores knowledge in A-V pairs.

4 - 32

Justification

M.1 has limited justification capability designed for an
exchange with the user. M.1 provides 'Why' justification.
'Why' will display the rule under consideration. It does not
fulfill the level of justification desired in XTRAM. However,
the capability desired can be integrated with the knowledge base.

Speed

M.1 is written in "C" and would find a solution in a
reasonable amount of time.

Friendliness

Developing a simple, goal-directed expert system with M.1
would be easy. M.1 provides a tracing capability and was
designed for individuals with no prior experience in knowledge
systems.

Conclusions

M.1 should not be used for XTRAM because the reasoning process
and the ability to store knowledge is limited.

4.2.7 Personal Consultant Plus (PC) by Texas Instruments

PC+ is a goal-oriented expert system and has the following features:

o Ability to interface with "external functions"
o Reasons primarily with backward chaining
o Knowledge is stored in A-V pairs
o Is not difficult to use

INGRES Interface

PC+ does not have an INGRES interface, but one can be added via "external functions".

Uncertainty

PC+ employs four "certainty" methods. They are: definitely-false to unknown to definitely-true (full), unknown to definitely-true (positive), unknown and definitely true or false (unknown), and definitely true or false (no value). They use a range of : -100 to 100 for 'full', (-100 <= X <= 100 ; X is a certainty factor), 0 to 100 for 'positive', (0 <= X <= 100; X is a certainty factor), -100 or 0 or 100 for 'unknown', (X is a member of the set {-100,0,100}; X is a certainty factor), and -100 or 100 for 'no value', (X is a member of the set {-100,100}; X is a certainty factor). The unknown range is -20

to 20 for 'full', (-20 <= X <= 20; X is a certainty factor), and
0 to 20 for 'positive', (0 <= X <= 20; X is a certainty factor).

PC+ manages degrees of uncertainty in four ways. First,
facts may be combineu by more than one rule. A combining
function blends the certainty factors. Second, compound
premises, combined by 'AND' or 'OR', may test uncertainty
factors. Third, an uncertain premise leads to an uncertain
conclusion. Lastly, rules themselves may be less than definite.
These rules are modeled after the MYCIN expert system.

Host Computer

PC+ is available on both symbolic and numeric computers. It
operates on: the IBM PC AT, TI Bus-Pro (IBM PC AT compatible)
and the TI Explorer.

Reasoning Process

PC+ is a backward chaining system with limited forward
chaining capability. A knowledge engineer can specify a rule to
be forward or backward chaining and the rules can be
hierarchically organized into frames. Each frame can be
dedicated to solving a particular problem.

Knowledge

PC+ stores knowledge in A-V pairs. Logically related A-V
pairs can be grouped together and in the frame hierarchy a child
frame inherits all the A-V pairs of its ancestor frames.

Justification

PC+ has a limited justification capability designed for a
dialogue with the user. PC+ provides 'How' and 'Why'
justification. 'How' lists the value of a requested attribute
and the rule(s) used to find it. 'Why' explains why information
is being requested, and cites the rule it is trying to use or
explains what the rule is about. It does not fulfill the level
of justification desired in XTRAM. However, the capability
desired can be built into the knowledge base.

Speed

PC+ is written in LISP and must perform "garbage collection".
Thus it would be slower on numeric computers, i.e. IBM PCs.

Friendliness

Developing a goal-directed expert system with PC+ would be easy. PC+ provides: LISP and knowledge base editors, tracing of the reasoning process, and the capability to save and replay consultation responses.

Conclusions

PC+ should not be used for XTRAM because the ability to store knowledge is limited. PC+ can access external functions, has certainty factors and is friendly. It can't run on the DEC Micro VAX, and must perform "garbage collection".

4.2.8 S.1 by Teknowledge

S.1 can be used to build goal-oriented expert systems and has the following features:

o Ability to interface with "external functions"
o Resides on a Micro VAX
o Reasons primarily with backward chaining
o Knowledge is stored in O-A-V triplets
o Finds solutions in a reasonable amount of time
o Is not difficult to use

INGRES Interface

S.1 does not have an INGRES interface, but one can be added via "external functions".

Uncertainty

S.1 uses certainty factors to handle uncertain information.
The certainty factors permit definitely-false to unknown to
definitely-true with a range corresponding to a closed interval
of real numbers between -1.0 and 1.0, ($-1.0 <= X <= 1.0$; where X
is a certainty factor). The unknown range is between -0.2 and
0.2, ($-0.2 <= X <= 0.2$; where X is a certainty factor).

Four general principles are used for combining two certainty
factors. First, once something is certain, it should not change.
Second, a combination should be commutative; it should not matter
which conclusion was made first. Third, no combination of
uncertain evidence should produce a result that is certain,
either definitely true or false. Finally, evidence of equal
strength but of opposite sign, should cancel and leave no effect.
These certainty factor combinations are modeled after the MYCIN
expert system.

Host Computer

S.1 is available on both symbolic and numeric computers, and
operates on: Symbolics, Xerox, DEC VAX and Micro VAX computers.

Reasoning Process

S.1 is a backward chaining expert system with a limited
forward chaining capability available through control blocks. A
control block specifies actions to be performed, the order in
which they are to be performed, and the conditions under which
they are to be performed. The control blocks provide the
knowledge engineer with limited control of the reasoning process.

Knowledge

S.1 stores knowledge in O-A-V triplets. Each triplet may have
multiple objects and each object may have multiple values with
associated certainty factors. In addition, the triplets and
values can be hierarchically organized.

Justification

S.1 has a limited justification capability designed for an
exchange with the user. 'What', 'How', and 'Why' justifications
are provided. 'What' provides a listing of values, i.e. the
color of the room is red. 'How' provides information about the
method used to determine a value: i.e., asked user or rule
invoked. 'Why' provides explanations about the inference process
that determined the value: i.e., information about control

4 - 39

blocks and rules. These explanation probes do not fulfill the
level of justification desired in XTRAM, although the level
desired can be built into the knowledge base.

Speed

S.1 is written in "C" and finds solutions in a reasonable
amount of time.

Friendliness

S.1 is a friendly expert system. S.1 has facilities to
trace the reasoning process, and edit and examine the knowledge
base.

Conclusions

S.1 could be used as the XTRAM expert system. S.1 can access
external functions, has certainty factors, resides on the Micro
VAX, uses O-A-V triplets, is reasonably fast, is written in "C"
and is friendly. Any missing items, such as justification, can
be programmed. The only drawback is that S.1 is primarily a
backward chaining tool.

4.3 Summary

Only two expert system tools, ART, and S.1, passed the XTRAM criteria, the results are contained in table 4-1. The other six tools failed one or more criteria, with the most commonly failed criteria being: compatibility with the DEC Micro VAX computer, friendliness, flexibility of the knowledge base, and control of the reasoning process.

Table 4 - 1: Comparison of XTRAM's Candidate Expert Systems

		ART	S.1	KEE	Kcraft	KES	IKE	PC+	M.1
H	Symbolics	x	x	x	x				
a	TI Explorer	x		x	x			x	
r	LMI	x		x			x		
d	Xerox		x	x					
w									
a	DEC VAX	x	x		x	x			
r	Micro VAX	x	x		x				
e	IBM PC		x(4)			x		x	x
	Language	LISP C(5,7)	LISP C	LISP	LISP	LISP C (5)	LISP	LISP	C
D	Data Base	P	P	P	P	P	P	P	P
a	Procedures	F	F	F	F	F	F	F	F
t									
a									
	Uncertainty	P	F	P	P	F	F	F	F
C	Forward chain	K	K,L	K	K		L	K,L	K,L
o	Backward chain	K	K	K	K	F	F	F	F
n	Breadth-first			K					
t	Depth-first			K (2)					
r									
o				(6)					
l									
F	A-V Pairs					x		x	x
a	O-A-V Triplet		x						
c	Frames	x		x	x		x		
t	Instantiation	F	F	F	F	L		L	
s									
	Friendliness	Good	Good	Good	Poor	ok	ok	Good	Good
	Justification	P	L	P	P	L	L	L	L
	Comments	1,3,8		1,3	1				

Legend: Poor < Ok < Good
    x: Has this feature
    K: Function can be controlled by knowledge engineer
    F: Function present
    L: Limited function present
    P: Function can be programmed

Comments: 1) Has facilities to compare several hypothetical situations
    2) Used by backward chaining mechanism
    3) The inference engine is easily modified
    4) Requires Unix board; delivery, not development version
    5) 'C' version is not completed
    6) Relies heavily on OPS5 and Prolog
    7) To be released in January 1987
    8) Manuals well written

5.0 Detailed Design Specification

The Detailed Design Specification of the proposed
hardware/software alternatives for the XTRAM system include one
hardware system, and two knowledge software systems. The hardware
system, as recommended in Section 3, is a DEC VAXstation II
workstation. The final two knowledge software alternatives to be
considered for XTRAM are ART by Inference, and S.1 by Teknowledge.

5.1 Detailed Specification of the Hardware Alternatives

The one hardware system chosen in the hardware analysis was the
VAXstation II workstation by DEC. Some of the features of the
VAXstation include: compatibility with all other VAX computing
environments, up to 9M bytes of memory, high resolution graphics,
Ethernet communications, multiwindowing, and multitasking.

The Hardware Functional Block Diagram (Figure 5-1) shows the
hardware components required for the integrated XTRAM/EPMIS system.
The 9M of memory and 159M bytes of disk space are required for
knowledge software development and are not necessarily required for
a deliverable knowledge system. The Ethernet communication package
is necessary for fast communications between the XTRAM and EPMIS
VAXs. A fast communication link is required for the many data
transfers and database accesses which will occur in the XTRAM
decision process. High resolution graphics will be used so both

5 - 1

XTRAM HARDWARE FUNCTIONAL BLOCK DIAGRAM

95MB
TAPE
SUBSYSTEM

WINCHESTER DISK

159MB

DEC VAXstation II
32 Bit Processor
9 MByte Memory

MASS STORAGE
SUBSYSTEM

QVSS
VIDEO
SUBSYSTEM

DEQNA
ETHERNET
COMMUNICATIONS

19"
MONOCHROME
GRAPHICS
MONITOR

3
BUTTON
MOUSE

LA 210
PRINTER
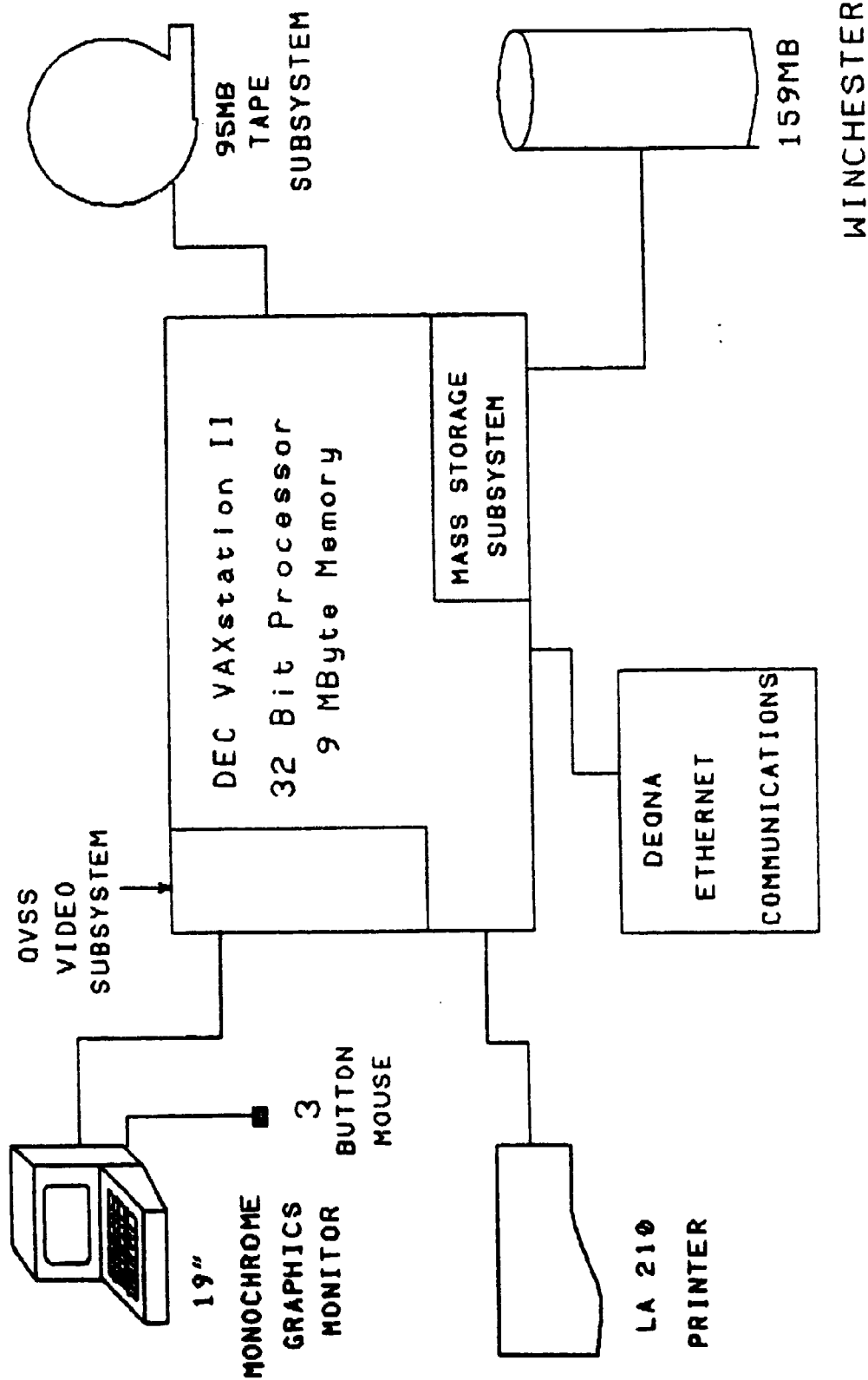
Figure 5-1

XTRAM and EPMIS can be viewed on the same monitor with no loss in resolution.

The prices and p :t numbers for the hardware components, and maintenance agreements associated with DEC and the VAXstation are listed in the following tables labeled hardware parts list and DEC Maintenance.

5.2 Detailed Specification of the Software Alternatives

The software required for the XTRAM system (Figure 5.2) consists of knowledge software, DEC system software such as compilers and editors, and INGRES database management software. Other software necessary for the XTRAM system include routines written by Delta which will perform numeric computations needed by the knowledge system, and also act as an interface to the EPMIS database.

5.2.1 DEC System Software

The DEC system software needed for the XTRAM system includes: a MICROVMS operating system, and a "C" compiler. The prices and part numbers, along with the license agreements are listed in the following table labeled DEC Software.

5.2.2 INGRES Database Management Software

INGRES Database Management Software is necessary in order to communicate with the EPMIS database. There are two INGRES software

HARDWARE PARTS LIST

ITEM	QUANT.	MANU.	DESCRIPTION	PRICE
1.	1	DEC	SV-LV55B-EK VAX STATION II	30,500.00
2.	1		MS630-BB 4MB RAM BOARD	3,500.00
3.	2		BNEC3C-10 10M ETHERNET CABLE	180.00
4.	1	DEC	RD54A-BA 159 MB DISK DRIVE	7,900.00
5.	1		KA210-AA DOT MATRIX PRINTER	1,595.00
6.	1		BC16M-6 PVC THINWIRE CABLE 6'	20.00
7.	2		H8223A T-CONNECTORS	15.00
8.	2		H8225-A	12.00
9.	2		DEST-AA STATION ADAPTER	275.00
10.	1		DEQNA-M QBUS ETHERNET CONTROLLER	1,975.00
11.	1		CK DEQNA-KB CABINET KIT	150.00
			TOTAL	$47,397.00

DEC MAINTENANCE

ITEM	DESCRIPTION	PRICE
1.	BASIC SERVICE FOR VAX STATION II	333.00
2.	BASIC SERVICE FOR MS030-BB	72.00
3.	BASIC SERVICE FOR DELNI-AA	10.00
4.	BASIC SERVICE FOR RD54A-BA	63.00
5.	BASIC SERVICE FOR LA210-AA	28.00
6.	HARDWARE INSTALLATION	1,354.50
7.	BASIC SERVICE Q4001-85	200.00
8.	BASIC SERVICE Q4A96-85	86.00
9.	BASIC SERVICE Q4810-85	73.00
10.	BASIC SERVICE Q4015-85	64.00
11.	BASIC SERVICE	141.00
12.	BASIC SERVICE QZD04-85	141.00
		$2,559.50

FUNCTIONAL BLOCK DIAGRAM OF REQUIRED SOFTWARE

MICROVMS OPERATING SYSTEM

```
                    ┌──────────────────────┐
                    │     KNOWLEDGE        │
                    │   SYSTEM             │
                    │   SOFTWARE;          │
                    │   INFERENCE ART      │────┐
                    │      or              │    │
                    │   TEKNOWLEDGE S.1    │    │
                    └──────────┬───────────┘    │
                               │                │
  ┌──────────────┐             │          ┌─────┴──────┐
  │   INGRES     │             │          │  EXTERNAL  │
  │   USER       │             │          │    "C"     │
  │   INTERFACE  │             │          │  ROUTINES  │
  │   SOFTWARE   │             │          └────────────┘
  └──────┬───────┘             │
         ┆                     │
         ┆            ┌────────┴──────┐
         ┆            │    VAX        │
         ┆┄┄┄┄┄┄┄┄┄┄┄┄│    "C"        │
                      │   COMPILER    │
                      └───────────────┘
         │
  ┌──────┴──────────┐
  │    XTRAM        │
  │  EXPERT SYSTEM  │
  │  /DATABASE      │
  │   INTERFACE     │
  └──────┬──────────┘
         │
  ┌──────┴──────┐
  │ INGRES/NET  │
  │  NETWORK    │
  │  SOFTWARE   │
  └──────┬──────┘
         │
  ┌──────┴──────┐
  │  DECNET     │
  │  ROUTER     │
  │  SERVER     │
  │ (ETHERNET)  │
  └─────────────┘
```
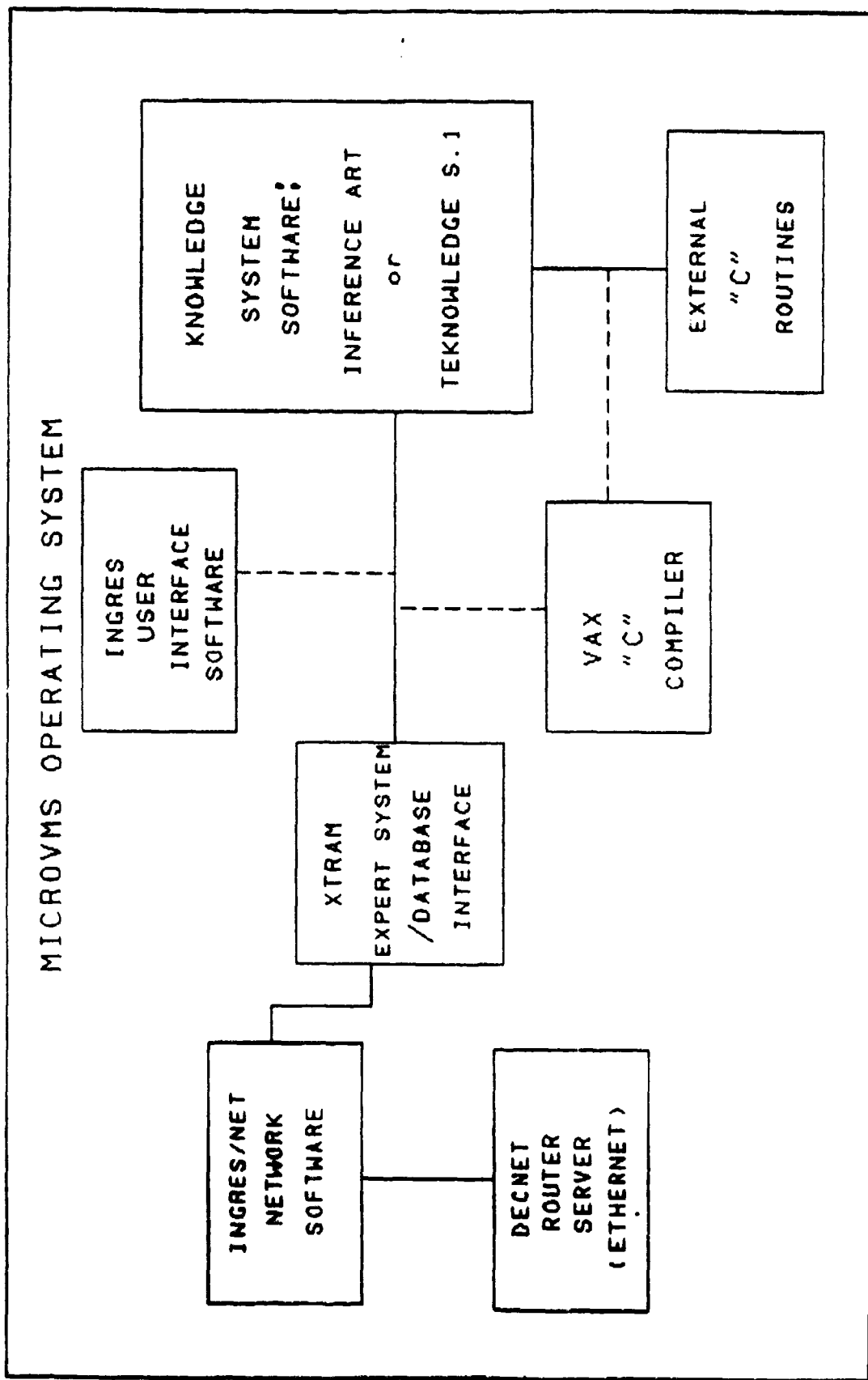
Figure 5-2

DEC SOFTWARE LIST

ITEM	QUANT.	MAN.	DESCRIPTION	PRICE
1.	1	DEC	Q4001-H5 MICRO VMS MEDIA & DOCUMENTATION	1,250.00
2.	1		Q4A96-H5 VAX STATION SOFTWARE MEDIA DOCUMENTATION	650.00
3.	1		Q4810-H5 VAX GKS/06 MEDIA & DOCUMENTATION	600.00
4.	1	DEC	Q4015-UZ VAX C/MICRO VMS LICENSE	709.00
5.	1		Q4015-H5 MEDIA & DOCUMENTATION	950.00
6.	1		Q4D04-UZ DEC NET LICENSE	500.00
7.	1		QD04-H5 MEDIA & DOCUMENTATION	600.00
8.	1		QZD04-UZ DECNET LICENSE	600.00
9.	1		QZD04-H5 MEDIA & DOCUMENTATION	141.00
			TOTAL	$6,000.00

packages needed in order to communicate with the EPMIS database on the MICROVAX. The first is INGRES User Interface Software which will be used in the building of database interface module. The second is INGRES/NET Network Software which will be used in the database data transfer process between the XTRAM and EPMIS VAXs. The prices and maintenance fees are listed in the following table labeled INGRES Software.

5.2.3 Routines Written by Delta

In addition to the previously mentioned software in the XTRAM software system, many routines will be written by Delta. These routines include a knowledge system/database interface, and routines which perform numeric and other computations as needed by the knowledge software.

5.2.4 Detailed Specifications of the Knowledge Software Alternatives

This section discusses the two alternatives, ART and S.1, for XTRAM in greater detail. The format used in section 4.0 was again used in this section with a pricing sub-section added.

5.2.4.1 ART by Inference

ART, a general purpose tool, is not geared to any specific problem, and has the following features:

INGRES SOFTWARE

ITEM	MAN.		LICENSE	MAINTENANCE
1.	INGRES User Interfaces		$ 750	$ 112.50
2.	INGRES Net:			
		uVAX	2,000	300.00
		VAXstation	600	90.00
			$ 3,350	$ 502.50
		TOTAL	$3,852.50	

o Ability to interface with "external functions"
o Ability to hypothesize
o Resides on a micro VAX
o Reasons with forward and backward chaining
o Knowledge is stored in frames
o Finds solutions in a reasonable amount of time
o Rules are compiled
o Is not difficult to use

INGRES Interface

ART does not have an INGRES interface, but one can be added by the knowledge engineer, via "external functions".

Uncertainty

ART does not have uncertainty-handling built-in; but the knowledge engineer can implement it via "external functions" or through a set of rules.

Host Computer

ART is available on both symbolic and numeric computers and operates on: the TI Explorer, LMI, DEC VAX and Micro VAX, or Sun computers.

Reasoning Process

ART has forward and backward chaining, and can reason about hypothetical situations. The forward and backward chaining can be

used separately, integrated or in combination. In addition, the backward chaining can be performed without "backtracking", meaning that it is not trapped in a stack-based sequence of operations.

The most important concept in ART is that of data-driven computation. In ART, the facts drive the rules. In a typical application, a rule is quiescent until a fact from the database matches one of the rule's patterns. When a fact or pattern match, several things may happen. If the rule contains multiple patterns, not all of which have been matched, ART may invoke backward-chaining rules to match the rule's remaining patterns. If all of the patterns have been matched, then ART creates an "activation" of the rule. Activations are sent to the agenda, which is a list of activations currently competing for an opportunity to act. ART evaluates this list and executes (fires) only the most important activation. After firing a rule ART revises the agenda, taking into account any changes in the database, and executes the most important activation in the revised agenda. This cycle repeats until interrupted or until ART discovers that the agenda is empty.

When a new fact is added to the database it is matched with patterns in all appropriate rules, moving them all closer to the agenda. This means that a rule enters the agenda only by having its last remaining pattern matched by a new fact. A newly asserted fact is the only factor that can fire a rule. This process constructs a chain of inferences from an initial set of known facts to some final

conclusion. Each time a rule fires, new facts are generated, which in turn becomes the basis for further inferences. Eventually this step-by-step process creates a set of facts that together form a useful conclusion. This process of building a chain of inferences from facts to conclusions is called "forward chaining".

It is also possible to identify a conclusion and work backward along a chain of inferences in search of known facts that would support the conclusion (backward chaining). Backward chaining in ART is generally used to lend support and assistance to the forward chainer, and in doing so finds the needed facts for partially-matched rules. The program identifies needed facts (desired conclusion) by generalizing from the patterns of partially-matched rules in the join net.

In addition, there are two typical applications of the backward chaining process that are of special interest: first, if a diagnostic program used forward chaining to interrogate a user, it would demand information about every missing piece of information whether relevant or not. A backward chaining rule, however, would ask for only that information of interest to one or more partially matched rules. As each question was answered, succeeding questions would become more and more pertinent as the set of applicable rules diminished. Secondly using backward chaining keeps the database as small as possible. In many applications there is no need to generate certain kinds of facts until they are actually needed.

Thus, although ART has backward chaining capability, its primary role is to supply facts to the forward chainer and to keep the database as small as possible.

ART also has the capability to pursue competing hypothetical pathways to a goal. It does this by building a tree-like or net-like structure of related viewpoints. A viewpoint is a collection of facts, which are visible to ART from a particular point of view. This point of view might refer to a situation as it existed at a particular time, or to a situation resulting from specific hypothetical assumptions, or to both at once, depending on the application. For example, if ART were planning a route down a path and encountered a fork in the trail, it would hypothesize two new viewpoints, one in which it takes the left fork, and one in which it takes the right fork. (See Figure 5-3.) ART would then continue down both paths and if it encountered further forks it would generate further hypothetical viewpoints. (See Figure 5-4.) When reasoning within one of these viewpoints, ART would be able to "see" all the assumptions leading to that specific situation. Other assumptions, leading to other situations, would be invisible to it. ART would operate only on the facts and assumptions pertinent to that hypothetical world.

In addition, ART can also reason about a situation that changes through time, such as the changing positions of the pieces in a game of chess. With each succeeding move the board positions change.
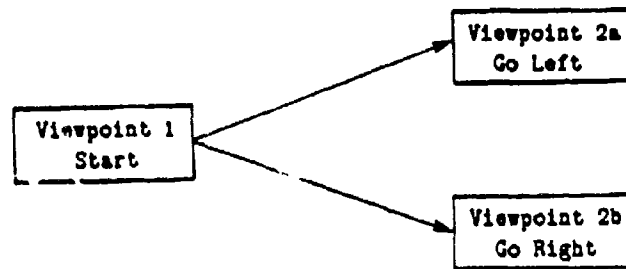
FIGURE 5-3
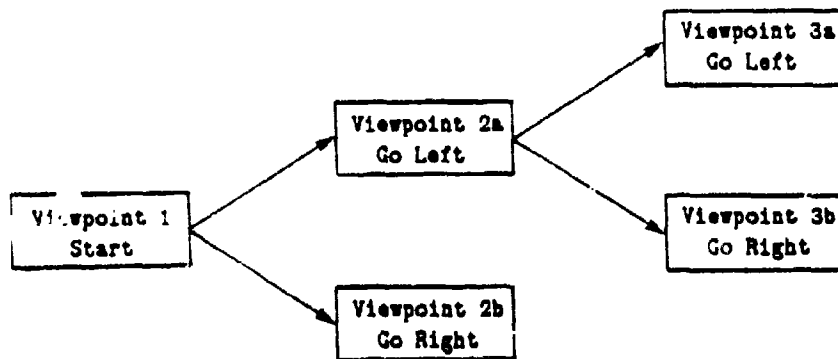
FIGURE 5-4

ART could construct a chain of viewpoints reflecting the state of the board after each move in a game. Such a viewpoint chain would represent the history of a game. When reasoning about any particular board position, ART would see only the facts pertinent to that state of the board or visible from that viewpoint.

This hypothetical reasoning capability would give XTRAM ability to answer 'why', 'why not' and 'what if' questions.

Knowledge

ART uses rules, facts, and schema to represent knowledge, and the knowledge embodied in the rules is used to manipulate the knowledge in the database.

A fact, in ART, consists of two parts; a proposition and an extent. The proposition is a fact expressed in words or equations. For example, some propositions might be "the car is red", "the car is a Ford", and "the car is paid for". Each proposition is a fundamental piece of information about the car. The second component of a fact is its extent and describes the circumstances in which the fact has meaning. For example, the proposition "the car is red" depends on a specific time frame. Correctly stated, the proposition should be "the car is red at this time". The extent of a fact describes the viewpoints in which the proposition is valid.

Thus, the proposition is the fact, and the extent is the time frame in which it is true.

ART also uses schema to represent knowledge, and is a collection of facts that represents an object or class of objects that share certain properties, such as a person, place or thing. This is analogous to a dictionary entry in which a word is followed by a series of phrases that describes its meaning. The phrases for a schema are a series of slots which represent the collection of facts known to be true of the schema. In addition, schemata can inherit characteristics of more general schemata, and ART provides two inheritance relations for this purpose. These relations take the form of special slots and contain the names of one or more related schemata. The two standard inheritance relations are: "is a ..." and "instance of ...". For example, within a dog schema the relationship "is a mammal" links two general concepts, and "Fifi is an instance of dog" represents one of something, a particular dog, but not all dogs. (See Figure 5-5.)

The two standard inheritance relations also have inverses, but are not inheritance relations. The inverse of "is a ..." is "kinds ...", and a list of the "kinds of mammals" would include dog, but would also include many other schema names. (See Figure 5-6.) Also, the inverse of "instance of ..." is "has instances ..." and dog would "have instances" Fifi, etc. (See Figure 5-7.) ART recognizes that inheritance links come in natural pairs and

```
┌─────────────────────┐
│      mammal          │
└─────────────────────┘
          ▲
          │
         is a
          │
┌─────────────────────┐
│       dog            │
└─────────────────────┘
          ▲
          │
      instance of
          │
┌─────────────────────┐
│       Fifi           │
└─────────────────────┘
```
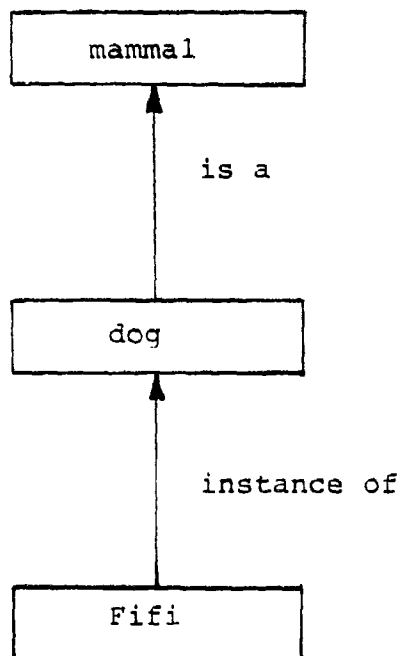
Figure 5-5: Example of Inheritance and Relationships

```
                    +------------------+
                    |      Mammal      |
                    +------------------+
                   /                    \
            Kinds /                      \ Kinds
                 /                        \
    +------------------+          +------------------+
    |       Dog        |          |     Dolphin      |
    +------------------+          +------------------+
```
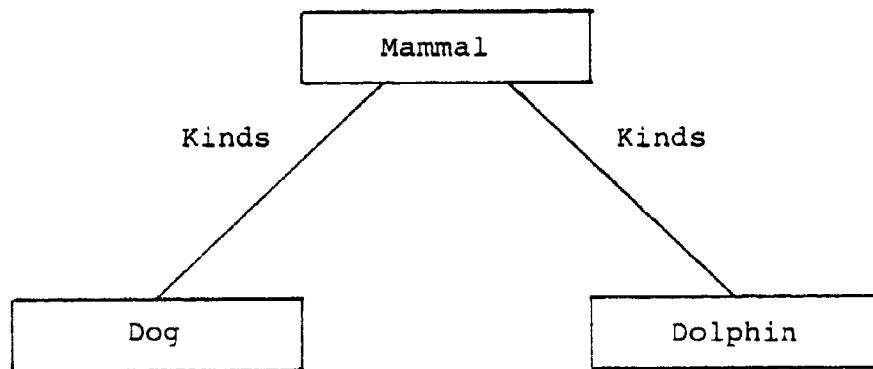
Figure 5-6: Example of "Kinds ..."

```
                    +------------------+
                    |       Dog        |
                    +------------------+
                   /                    \
    has instance  /                      \ has instance
                 /                        \
    +------------------+          +------------------+
    |       Fifi       |          |      Rover       |
    +------------------+          +------------------+
```
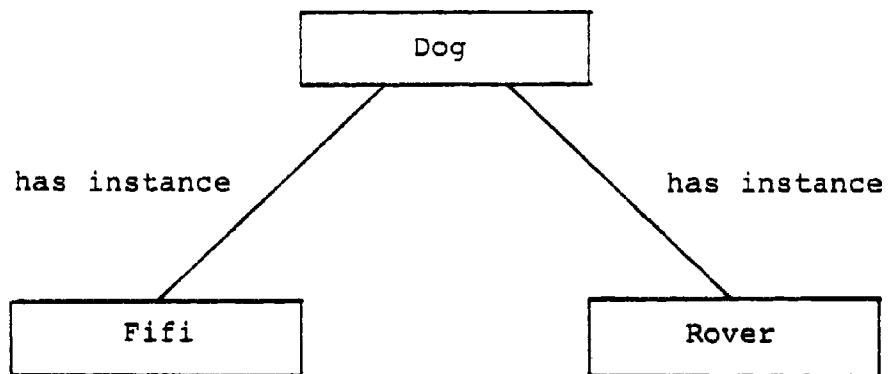
Figure 5-7: Example of "has instance ..."

automatically creates the missing half of the pair. The knowledge engineer need only enter half of the link.

Lastly, ART's schema system is defined in terms of itself. The kernel schemata define the behavior of slots and relations, and permits knowledge-engineer-defined schemata, which define special relations, to be attached. Thus, the slots defining the appropriate behavior can be inherited automatically.

Justification

ART does not have any built-in justification; but it can be implemented through "external functions" and rules. Also, the effort to develop 'what if' and 'why not' justification can be reduced by ART's ability to reason about hypothetical situations without changing the original facts or deductions.

Speed

ART is considered to be one of the fastest expert systems. The speed is achieved by compiling the rules and knowledge base instead of interpreting them. In addition, two other methods were used to achieve a fast expert system. First, ART does not perform "garbage collection"; it uses a dynamic memory management system which deallocates memory when it is no longer needed. Second, the pattern matching structure has the ability to join rules from the

right in addition to the standard capability of joining rules from the left. A join is a mathematical operation that identifies commonalities among patterns in the database. In general, to speed up computation the joins should be as general as possible. Thus, ART's speed comes from compiling rules, eliminating "garbage collection" and performing joins from the right in addition to the traditional joins from the left.

Friendliness

Developing an expert system with ART would not be difficult. The knowledge engineer has to work with the underlying language, LISP or "C", only if procedural functions are needed to interface with INGRES. The knowledge engineer interface, ART Studio, permits browsing, editing and debugging of the knowledge base. ART Studio is based on a menu system and menus can be selected using either the mouse or the keyboard. The menus provide access to the data base and monitoring of program execution.

Pricing

The "C" version of ART is being introduced at a reduced price of 45,000 dollars, until 31 December 1986. ($65,000 after 31 December 1986. Price is negotiable, however.) It includes 5 days of Inference's knowledge engineer consulting time, 2 sets of ART documentation (Reference manual, 3 tutorial manuals, primer and an

Artist manual.) and 90 days of maintenance and technical support
(Inference hotline, new ART versions and documentation updates). An
annual maintenance and technical support agreement is also available
for $7,500 and commences when the 90 day agreement runs out. (The
standard Inference product license agreement is contained in
appendix B.)

A two week training course is recommended by Inference and is
priced at $2,500 per person. (See letter in Figure 5-8.) The first
week is used to introduce the knowledge engineer to ART, and the
second week is used to instruct the knowledge engineer on how to use
the viewpoints feature of ART. Inference normally suggests the
knowledge engineer should take the first week of the course, work
with ART for a month and then take the second week of the course.

Lastly, the multiple copy license for the delivery version of
ART for the developed expert system costs for copies 1 - 10, $8,000,
copies 11 - 50, $7,000, and copies 51 - 100, $5,000. (See letter in
Figure 5-9.)

5.2.4.2 S.1 by Teknowledge

S.1 can be used to build goal-oriented expert systems and has
the following features:

o Ability to interface with "external functions"
o Resides on a Micro VAX
o Reasons primarily with backward chaining

5 - 21

inference

To:
Chuck Ziegler
Delta Information Systems
Horsnam Business Center
Building 3
300 Welsh Road
Horsham, PA 19044

213 417-7997

NO.	1330-OH
Date	15 October 1986
P O Deadline	31 December 1986

The price shown in this Quotation shall not be increased for thirty (30) days from the date shown above.

License Agreement must be executed and returned to Inference Corporation prior to receipt of software. Terms: Net 30 days from invoice date

Item	Description	Price	Amount
	ART - AUTOMATED REASONING TOOL		
	• ART: (copy 1 "C" Version/Delivery before 12/31/86) 5 Days Knowledge Engineering 2 Sets of ART Documentation 90 Days Maintenance & Technical Support - Full access to Inference Hotline - New Version & Documentation Updates	545,000.00	
	TRAINING: Per person/per week Week One - Introduction to ART Week Two - Viewpoints	2,500.00	
	KNOWLEDGE ENGINEERING: Per day, plus expenses (Charged in 1/2 day increments 4 hours = 1/2 day)	1,500.00	
	MULTIPLE COPY DISCOUNT SCHEDULE:		
	Copies Price KE Included 2 - 5 45,000 4 days 5 - 10 38,000 3 days 11 - up 29,500 2 days		
	ANNUAL MAINTENANCE & TECHNICAL SUPPORT: (Commences 90 days after shipping, billed quarterly, payable in advance).	7,500.00	
	• P.O. and signed license agreements must be received prior to 31 December 1986 to qualify for discount.		

By *Donald R. Gammon*
Donald R. Gammon/VP Sales & Marketing

FIGURE 5-8: DEVELOPMENT ART PRICE QUOTE

Inference

_____

October 23, 1986

Mr. Charles Ziegler
Delta Information Systems
Horsham Business Center
Building 3
300 Walsh Road
Horsham, PA 19044

609 985-0505

Dear Chuck:

I am delighted to hear Delta is preparing to acquire a copy of
the Automated Reasoning Tool, A.R.T. Inference Corporation
is anxious to provide Delta with the best support available in
the expert systems marketplace.

In that regard, let this letter document my recommendation for
the preferred development environment. You have specified
A.R.T./C Code to be delivered to Delta by January 1987. That
requirement is acceptable to Inference. Therefore, Inference
recommends your order specify A.R.T./C Code rather than
A.R.T./Lisp Code, with the target hardware being a Dec A.I.
Workstation.

Also, be aware that your runtime modules can be deployed in
A.R.T. C. Code and will be over 99% garbage free. The following
discount schedule will apply to the runtime modules shipped by
Delta Information Systems:

Copies	Pricing
1 - 10	8,000
11 - 50	7,000
51 - 100	5,000
101 - 200	3,000
201 - 500	2,000
501 - up	1,500

In closing, please feel free to contact me at the Marlton Office
should you have further concerns or questions.

Regards,

Dennis Hartigan

Dennis Hartigan
Inference Corporation
Marlton, New Jersey

DH x.s

FIGURE 5-9: DELIVERY ART PRICE QUOTE

5-23

o Knowledge is stored in O-A-V triplets
o Finds solutions in a reasonable amount of time
o Is not difficult to use

INGRES Interface

S.1 does not have an INGRES interface, but one can be added via
"external functions".

Uncertainty

S.1 uses certainty factors to handle uncertain information.
The certainty factors permit definitely-false (-1.0) to unknown to
definitely-true (1.0), ($-1.0 < X < 1.0$; X is a certainty factor).
The unknown has its own range from -0.2 to 0.2, ($-0.2 < X < 0.2$; X
is a certainty factor).

Certainty factors are combined using four general principles.
First, once something is certain, it should not change. When a fact
is either definitely-false (-1.0) or definitely-true (1.0) no new
evidence pertaining to that fact will be used. Second, a
combination should be commutative; it should not matter which
conclusion was made first. Third, no combination of uncertain
evidence should produce a result that is certain, either definitely
true or false. Finally, evidence of equal strength but of opposite
sign, should cancel and leave no effect. For the last three, three
rules apply:

o If x and y are both greater than 0 but less than 1.0.

The cumulative certainty factor is the value of x
plus the value of y minus the product xy. Specifically:
x + y - xy.

o If x and y are both less than 0 but greater than -1.0.
The cumulative certainty factor is the value of x
plus the value of y plus the product xy. Specifically:
x + y + xy.

o If x and y are of opposite sign and neither is equal
to 1.0 or -1.0.
The cumulative certainty factor is a quotient. The
numerator is the absolute value of x plus the absolute
value of y. The denominator is 1.0 minus the minimum
of the two absolute values. Specifically:
$(|x| + |y|)/(1 - min (|x|,|y|))$.

The conclusion statement that appears in a rule is comprised

of individual conclusions and each as a certainty factor associated

with it.   If no explicit certainty factor is specified, S.1 uses 1.0

as a default.   The certainty factor in the conclusion represents the

degree of belief with which the conclusion can be asserted, if the

premise of the rule is found to be true with a certainty of 1.0.

For a rule to succeed, the certainty factor premise must be

greater than 0.2.   If the premise of the rule has a certainty of

1.0, the conclusion is made as explicitly stated in the rule.   If

the premise of the rule has a certainty that is less than 1.0, the

certainty of the conclusion is lowered.   This is done by multiplying

the certainty factor in the conclusion with the certainty factor of

the rule's premise.

The construction of the premise affects the determination of

its certainty factor.   If the premise is a conjunction of

expressions, its certainty factor is the minimum of the certainty factors of all the conjuncts. If the premise is a disjunction of expressions, its certainty factor depends upon the certainties of the individual expressions. If one has a certainty factor of 1.0, it is used. If none have a certainty greater than 0.2, then -1.0 is used. Otherwise, the result is the maximum certainty of the expressions.

These certainty factor rules are modeled after the MYCIN expert system.

Host Computer

S.1 is available on both symbolic and numeric computers, and operates on: Symbolics, Xerox, DEC VAX and Micro VAX computers. In addition, one of Teknowledge's customers has ported a delivery S.1 to the IBM PC. It is not, however, a supported product. (A Unix board was added to the IBM PC in order to make it work.)

Reasoning Process

S.1 is a backward chaining expert system with limited forward chaining capability available through control blocks. A control block specifies actions to be performed, the order in which they are to be performed, and the conditions under which they are to be performed. Specifically, they can create instances of classes (See

Knowledge), determine attributes, display text, and invoke other control blocks. The forward chaining capability would be achieved with control blocks. Backward chaining is already built into S.1 and can be started using a control block.

Knowledge

S.1 stores knowledge in O-A-V triplets. Each triplet may have multiple objects and each object may have multiple values with associated certainty factors. These are defined as: A-On-(V/cf)m, and are known as classes of objects. Classes describe entities, events and concepts about which the system has knowledge. The attributes represent properties of classes. Mathematically, they map an n-tuple of classes (the On part) to a set of values (the Vm part). Each fact gathered by the system is represented by the value of an attribute applied to an n-tuple of class instances. (An instance represents individual members of a set described by a class. For example, with ROOM and HOUSE as names of classes, a particular house, HOUSE #1, is an instance. This house has four rooms, ROOM #1, ROOM #2, ROOM #3, and ROOM #4, each of which is an instance of class ROOM.) Thus an attribute of HOUSE could be "is the house single-story?", true or false.

Justification

S.1 has a limited justification capability designed for an exchange with the user. 'What', 'How', and 'Why' justifications are provided. They are designed for the user to query why a particular question was asked, such as the rule which prompted the question, etc. They do not fulfill the level of justification desired in XTRAM, although the level desired can be built into the knowledge base.

Speed

S.1 is written in "C" and does not perform any "garbage collection". Thus S.1 will not shut-down in mid-application and will find a solution in a reasonable amount of time.

Friendliness

S.1 has facilities to trace the reasoning process, and edit and examine the knowledge base.

Pricing

S.1 is available for 25,000 dollars. (See letter in Figure 5-10.) It includes: the development software, documentation (Users

Ronald K. Goldstein

October ?? ????

Mr. Chuck Zeigler
Delta Information Systems
Horsham Business Center
Building ??
300 Welsh Road
Horsham, Pa. ?????

Dear Mr. Zeigler

Teknowledge, Inc. is pleased to provide you with a quote for
MicroVAX based S.1 expert system software. S.1 is available for both
VMS and Unix (Ultrix) operating systems and can be delivered on either
cartridge or disk media.

Teknowledge offers the initial copy of S.1 and the following
additional items for a total of $25,000 for the MicroVAX version of
S.1

 o S.1 expert system development software for your work-
  station.
 o S.1 Documentation (Users Guide, Reference Manual and
  Sample Knowledge Systems).
 o The S.1 **packager** used to build and field delivery
  systems.
 o One week of Knowledge Engineering Methodology(KEM)
  training.
 o Two weeks of S.1 training; and
 o One year of telephone hotline support and S.1 product
  updates.

Additional development copies of S.1 are available at $25,000
through the 5th copy and decrease to $20,000 thereafter.

Delivery (or run-time) licenses of S.1 applications are priced at
$2,000 for the first through the 30th copy and $?,??? per copy for
the 31st through the 100th copy. Teknowledge is open to working with
Delta Information Systems in working out special pricing for
situations involving a commitment to purchase of development
and/or delivery copies of S.1

If you have any further questions please do not hesitate to call

Sincerely,
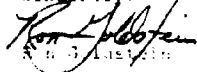
R. K. Goldstein

FIGURE 5-10: S.1 PRICE QUOTE

guide, reference manual and sample knowledge systems), packages used
to build and field delivery systems, one week of knowledge
engineering methodology training, two weeks of S.1 training, and one
year of telephone hotline support and product updates. Additional
development copies are: $25,000 for copies 1 through 5, and $20,000
for 6 or more copies.

The delivery licenses are available for $3,000 for copies 1
through 30, and $2,700 for copies 31 through 100. (The standard
Teknowledge software license agreement is contained in appendix C.)

5.2.4.3 Comments

Of the two candidates for XTRAM, ART is clearly the more
powerful. It can reason with both forward and backward chaining;
S.1 has a limited forward chaining capability. (Forward chaining is
more appropriate to XTRAM, it is used to find the number of ways to
link two nodes in a network, see section 4.1.) ART has a clearly
defined and rich knowledge storage capability; S.1's is more
primitive. ART can be easily programmed to handle 'what if' and
'why not' questions by hypothesizing; S.1 has to have this feature
added via its knowledge base, not necessarily an easy task. Though
both are written in 'C', and neither perform "garbage collection",
ART is faster because its rules are compiled; S.1's rules are
interpreted. Also, although S.1 has certainty factors (ART does
not), ART might still be favored because the knowledge engineer can

chose a certainty algorithm tailored to the application and program
it into ART.

In S.1's favor, S.1 is cheaper and has been in use for a longer
period of time.

In conclusion, ART is a better choice depending on the
importance of price.

6.0 EPMIS Compatibility Plan

One of the two requirements necessary to ensure XTRAM
compatibility with EPMIS, is hardware and software compatibility
with DEC VAX hardware. This requirement has been stressed by the
NCS in order to make the future XTRAM migration to the EPMIS VAX as
easy as possible. The other requirement involves the availability
of INGRES database management system network software. The INGRES
network software would be necessary to ensure an efficient database
access between the XTRAM VAX and the EPMIS VAX. With XTRAM
implemented on a DEC VAXstation II, both of these requirements would
be satisfied.

6.1 Hardware Interface

The XTRAM/EPMIS system integration plan is shown in Figure 6.1.

The hardware integration task would not be difficult since the
MicroVAX and VAXstation are very compatible. An Ethernet cable
would be the most efficient way to physically connect the two VAXs,
providing a transmission rate of approximately 10M bits per second.
DECnet network communications protocol would be used to network the
VAXs.

6.2 Software Interface

6 - 1

XTRAM INTEGRATION PLAN

XTRAM/EPMIS
HIGH RESOLUTION
GRAPHICS MONITOR

XTRAM
VAXstation II

- EXPERT SYSTEM
- EXPERT SYSTEM /DATABASE INTERFACE
- INGRES/NET
- DECNET

EPMIS MICROVAX

- EPMIS
- EPMIS DATA BASE
- INGRES/NET
- DECNET

ETHERNET COMM LINK
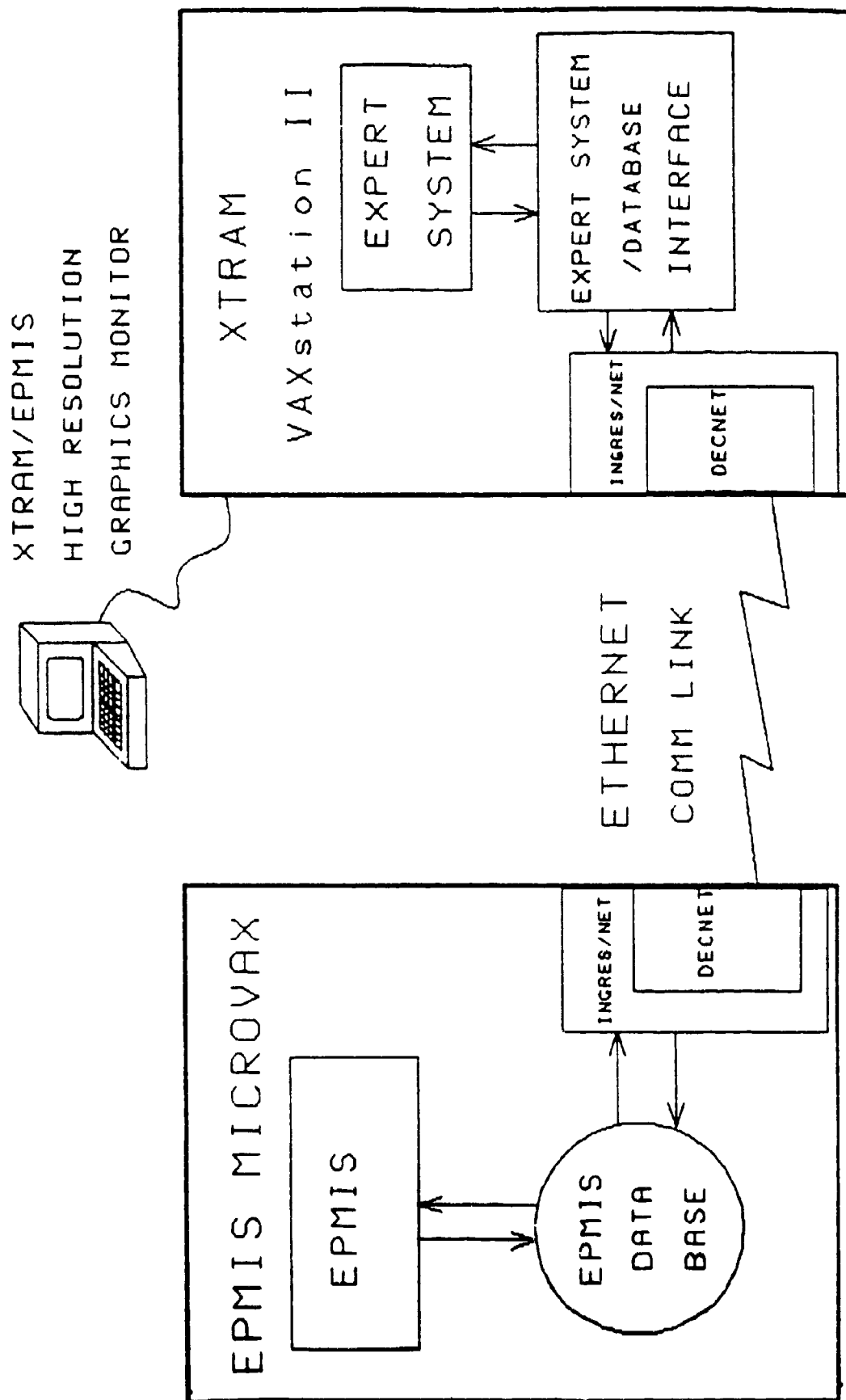
Figure 6-1

Virtually all of EPMIS is comprised of and managed by the INGRES database management system, including the EPMIS database which XTRAM must access frequently in the resource allocation process. The EPMIS database not only includes the resources that will be used by XTRAM, but also the actual service requests which are inputs to XTRAM. It is for this reason that for the most efficient database access, it is necessary that INGRES software is used to access the EPMIS database.

The XTRAM/EPMIS software interface plan to access the EPMIS database and retrieve the data needed by XTRAM requires the acquisition of INGRES/NET network software along with INGRES User Interface software. The INGRES User Interface software would be used to access the EPMIS database by embedding INGRES database query statements in a procedural language such as "C" or FORTRAN. Once the data has been retrieved, it would be manipulated by the XTRAM/EPMIS interface for processing by the expert system. The INGRES/NET network software is necessary for the most efficient and most transparent INGRES data transfer between the EPMIS and XTRAM computers.

The XTRAM software module consists of the expert system and a software interface which would enable the expert system to communicate with the EPMIS database. The expert system interface would consist of two parts. The first part would be the database interface described above which would query the database and put the

6 - 3

retrieved data into programming variables. The second part of the
interface manipulates these variables so that they could be used by
the expert system. The variables would then be passed to the expert
system for processing.

The software tools necessary for an EPMIS/XTRAM integration would
include: DECnet network software, a "C" compiler, INGRES/NET network
software, and INGRES User Interface software.

6.3 User Interface

The XTRAM user interface controls how XTRAM will appear to the
user, and how XTRAM would be accessed and used once it is integrated
with the EPMIS VAX. When XTRAM is connected to the EPMIS VAX,both
EPMIS and XTRAM could be accessed from the XTRAM terminal which
would be connected to the XTRAM VAX. In essence, the RAO would have
only one terminal in front of him with access to both EPMIS and
XTRAM. One possible XTRAM/EPMIS user interface setup is shown in
Figure 6-2. Using this type of XTRAM setup would satisfy the NCS's
desire for a minimal hardware configuration and a virtually
transparent XTRAM access from the EPMIS system.

A POSSIBLE XTRAM/EPMIS USER INTERFACE SETUP

19" HIGH RESOLUTION MONITOR

EPMIS
MAIN MENU

1. Emergency Procedures

2. Emergency P.O.C.s

3. Damage Assesment

...etc.

XTRAM
MAIN MENU

1. ....
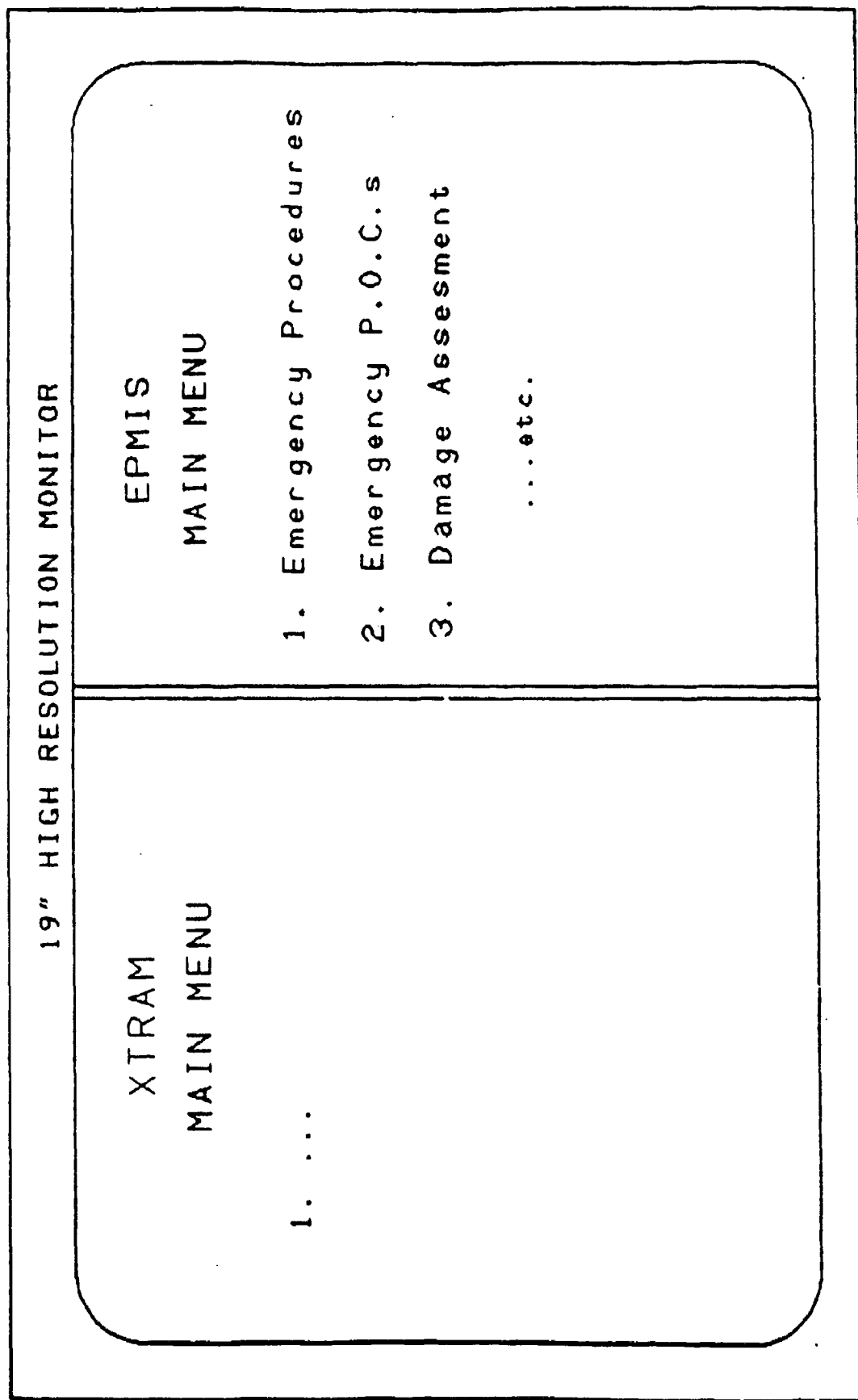
Figure 6-2

APPENDIX   A

DESCRIPTIONS OF EXPERT SYSTEMS

ART 3.0 PRODUCT BACKGROUNDER

The challenge for today's expert system tool companies will be to provide commercially viable products that will help corporations transform the enormous amounts of data already in their databases into useful information with which to manage their companies. To do that, companies such as Los Angeles-based Inference Corporation are developing expert system tools to automate human reasoning so that computers can perform professional problem-solving functions.

Expert systems accomplish professional problem-solving through the intensive use of knowledge rather than the carefully prescribed procedures used in traditional software programming. Inference has released its expert systems technology to industry and commerce through its product, the Automated Reasoning Tool^{TM} (ART^{TM}). ART supports the development of both decision support and decision-making software applications which help the white collar workforce achieve greater productivity.

INTRODUCING ART 3.0

ART 3.0 is an enhanced version of the Automated Reasoning Tool^{TM} (ART^{TM}), a software tool from Inference Corporation for helping developers create expert systems. ART 3.0 provides significant overall performance improvements over previous versions of ART and also broadens the market for the commercial use of expert systems applications. Major features now available with ART 3.0 are:

o   More than 99 percent garbage-free

o   Optimized join topology

o   Procedural object-oriented programming

o   File compilation capability

With this enhanced version, ART 3.0 is available on widely-used, general-purpose workstations from Sun Microsystems^{TM} (the Sun-3^{TM} workstation) and Digital Equipment

Inference

Corporation (the Digital VAX^{TM} line) as well as dedicated Lisp machines (such as Symbolics^{TM}, TI Explorer^{TM}, LMI, etc.) ART 3.0 will be available on the DEC^R VAX products running under VAX-Lisp and VMS^{TM}, and will be available on the Sun-3 workstation under Sun Common Lisp and UNIX^R.

ART 3.0 increases performance for both expert systems built in ART and for the development process itself.

FEATURES OF ART 3.0

ART 3.0 includes three key feature enhancements designed to speed its performance efficiency and make it a more robust tool for the development of commercial expert system applications. The three key feature enhancements, are the elimination of garbage collection, the addition of a unique feature called "joins-from-the-right", and an improvement to ART's rule-based programming environment called procedural attachments. Each of these is explained in more detail here.

ELIMINATION OF GARBAGE COLLECTION

Improved memory management makes ART 3.0 at least 99 percent garbage-free. By changing the way the program manages memory, ART 3.0 speeds up processing. Lisp programs usually have to divide their resources running programs and searching for available storage. Searching for storage is called garbage collection.

In the past, an AI program might arbitrarily shut down in mid-application to make a garbage run if it were running low on memory space. In commercial situations, such as financial service applications where programs must keep running continuously, this unpredictability has made expert systems software with garbage collection potentially unreliable.

ART 3.0 doesn't require garbage collection because ART automatically allocates and deallocates memory. Programmers need not worry about arbitrary shut-downs or performance degradation while the program is collecting garbage because they control how much memory is

Inference

available at all times. This new performance and reliability increases the range of practical applications that can be built in ART. It is particularly important in a real-time environment. The conversion of garbage collection to conventional programming practice is an important step in the commercializing of expert systems technology.

GENERALIZED JOIN TOPOLOGY

ART 3.0 uses a new pattern matching structure that saves computational time and allows the computer to work up to 100 percent more efficiently and more (in some applications).

ART 3.0 has a "generalized join topology" that lets the program join rules in its database from the left, as has been traditional in expert systems reasoning tools, but now, also from the right, which is an AI first. A join is a specific mathematical operation that results in identifying essential commonalities among complex patterns in the database. Each join represents a step in a complex match which, in turn, is a step in the problem-solving process.

The key to speeding up computation is to make joins as general as possible so there are fewer of them required to solve a particular problem. In traditional AI reasoning, tools that join from the left lead to information organized in the form of a sequential pattern. In ART 3.0, however, fewer joins are needed to solve a problem because ART generalizes the joined information into a tree-like structure, using pattern matching from the right, as well as from the left. In more descriptive terms, Inference has invented a way to integrate rule-based processing, which works most efficiently on mathematical relations that have a large number of arguments, with schema-based representation, which uses only binary relations.

By generalizing the rules pertaining to joins, ART 3.0 reorganizes the allocation of memory. This saves computational time and reduces the number of cycles needed to come up with an answer.

Inference

PROCEDURAL ATTACHMENT FEATURE ADDED TO OBJECT-ORIENTED PROGRAMMING

A procedural attachment feature has also been added to ART. Procedural attachments define a specific function in object-oriented programming used to attach active value to objects. The feature of procedural attachments is further enhanced with the implementation of a new concept called "multi-methods" (see ART 3.0 Features backgrounder for complete description). The use of multi-methods in ART is the first appearance of this technology in a commercial expert system tool product.

The addition of procedural attachments strengthens ART's object-oriented programming capability. ART is the only expert system tool to combine both rule-based and object-oriented programming in such a strong feature set.

For example, a programmer using multi-methods can define a "print" function that prints any part of a document on any particular device. Such a program is easier to write using an object-oriented system where you can associate the print command with an object type such as a line printer or a laser printer. The rule-based capability lets programmers identify a variety of types rather than just one. A device can be added to the rule without changing the structure of the program because in rule-based processing it doesn't matter what order a rule is entered in the program. In the past, the structure of a purely object-oriented program made it impossible to add new devices without rewriting the original code.

FILE COMPILATION

ART 3.0 also includes file compilation. Files of rules in source code can be compiled into binary files. This has two major benefits: speed and marketing flexibility. Binary files load 10 tir·· faster than source files and customers can deliver applications in binary files, thereby

Inference

protecting their proprietary source code. In ART 2.0, source files were loaded into an ART image. Compilation took place during loading, a significantly slower process.

ART 3.0 also includes the interactive graphics capabilities Studio^{TM} and ARTIST^{TM}. Both tools help make ART's applications accessible to users and developers who do not have extensive artificial intelligence backgrounds. Studio is designed to help the knowledge engineer construct an expert system and monitor its progress. ARTIST is an interactive tool used by ART programmers for building user-interfaces for their expert sytems.

PRICING AND AVAILABILITY

ART 3.0 will be available for Symbolics, Lisp Machines, Inc., Explorer, and DEC VAX (under VAX-Lisp) machines and for the Sun-3 workstation (under Sun Common Lisp). Pricing is $65,000 for the first copy and $45,000 for copies two through five. All current ART 2.0 users having active maintenance contracts will receive version 3.0 as an upgrade.

Inference supplies commercial expert system development tools, applications consulting and training to the aerospace, manufacturing and financial industries.

\# \# \#

Automated Reasoning Tool, ART, Studio and ARTIST are trademarks of Inference Corporation.

Explorer is a trademark of Texas Instruments, Inc.

Sun Microsystems and Sun-3 are trademarks of Sun Microsystems Inc.

VAX and VMS are trademarks of Digital Equipment Corporation.

DEC is a registered trademark of Digital Equipment Corporation.

UNIX is a registered trademark of AT&T.

Symbolics is a trademark of Symbolics, Inc.

Inference

THE EVOLUTION OF ART:
MOVING INTO THE MAINSTREAM

Over the past several years, artificial intelligence has produced a flurry of interest. In reality AI is just now beginning to fulfill its promise. Today the commercialization of expert systems requires that this new software technology be deliverable in traditional procedural languages in addition to Lisp.

The investment dollars in AI will bear fruit when expert systems become part of the solution for mainstream applications of computers, such as manufacturing and financial services. Inference Corporation's new release of its Automated Reasoning Tool^{TM} (ART^{TM}) a powerful programming language for building commercial expert system applications, is a key step in the evolution of these more powerful, flexible expert systems programs.

THE EVOLUTION OF ART

ART was first introduced in August of 1984. Since that time ART has been recognized for its advanced automated reasoning capabilities such as Viewpoints, a feature unique to ART designed to do both hypothetical and time-based reasoning, and for deep integration between a rich feature set of AI programming paradigms including forward and backward chaining, etc.

In August of 1985 ART was evolved into an enhanced product called ART 2.0. ART 2.0 delivered the same robust expert system features but was equipped with much improved user access to the knowledge-base being built and was optimized for speedier performance.

Now, ART is evolving further to include several new technological breakthroughs designed to take a major step forward in commercializing this new technology.

As ART has evolved and stabilized technologically, the demand for expert systems has grown rapidly. Commercial customers are demanding expert systems which can be easily deployed into today's existing computer environment. To meet this need, Inference is delivering the latest

Inference

version of ART, ART 3.0, in a choice of languages. Initially, ART 3.0 will be available in the preferred AI development language, Lisp. Following will be ART 3.0 in C, designed to deliver efficiency in expert systems development and deployment as well as easy connectivity with popular computers installed commercially today.

FEATURES OF ART 3.0

ART 3.0 includes several features specifically designed to increase its performance and make it a viable language for building general applications for the office, manufacturing and general data processing markets.

For example, ART 3.0 in Lisp is at least 99 percent garbage-free, which means that it runs faster than traditional Lisp programs. Lisp programs usually divide their resources to run programs and search for available storage (i.e. memory management). A Lisp program slows considerably and even will shut itself off to seek out storage. This searching for storage is called garbage collection. ART 3.0 does not require garbage collection because ART automatically allocates and deallocates memory. With the program free to process without garbage collection, performance (both speed and efficiency in processing) increases substantially. Thus, by giving up one of the development features of Lisp, automatic garbage collection, Inference has taken a major step forward in the process of commercializing expert systems.

Inference has also invented a new way to solve problems that saves computational time and allows the computer to work more efficiently. ART 3.0 incorporates a "generalized join topology" that lets the program join rules in the database from the left, as is standard in AI reasoning tools, and from the right, an AI first. This "joins-from-the-right" feature delivers significantly increased performance.

SUMMARY OF NEW FEATURES

In joins-from-the-right, the structure of relational joins -- those generated by the ART rule compiler to determine the matches of a rule antecedent -- have been generalized from

Inference

sequential joins to an arbitrary binary tree of joins. This technique represents a qualitative improvement in the integration of rule-based processing and schema-based representation, and a fundamental improvement in rule compilation technology. ART is the first rule-based tool to provide this capability.

A PROCEDURAL ATTACHMENT FEATURE

A procedural attachment feature has also been added to ART. Procedural attachments define a specific function in object-oriented programming used to attach active values to objects. The feature of procedural attachments is further enhanced with the implementation of a new concept called "multi-methods" (see ART 3.0 features backgrounder for complete description). The use of multi-methods in ART is the first appearance of this technology in a commercial AI product.

The addition of procedural attachments strengthens ART's object-oriented programming capability. ART is the only expert system tool to combine both rule-based and object-oriented programming in such a strong feature set.

ART 3.0 IN C

For commercial users developing expert system computer applications, the "tool-of-choice" needs to deliver improved performance combined with ease of interfacing to existing computer environments. For these commercial users, the "tool of choice" need not to be tied to Lisp. While Lisp is a powerful exploratory development language, once a porgram like ART, originally developed in Lisp, becomes sufficiently stable, it can be translated to traditional programming languages like C or PL1 for greater run-time efficiency. The underlying AI technology of ART can be delivered without loss of functionality or robustness irrespective of the language in which ART runs.

The C language, one of the mainstream programming languages is fast and efficient in its processing power and is widely-known. Thus, C meets the requirements for preferred language in

Inference

which to deliver ART. Development of application in ART, wirtten in C allows the expert system to get closer to the underlying machine architecture, thus delivering enhanced speed and performance efficiency.

ART 3.0 in C also allows expert systems to be developed in the same language on the same machines in which the application will ultimately run. This leads to a better engineered, more stable application.

To deliver ART 3.0 in C, Inference solved a major AI performance challenge by eliminating the need for garbage collection. Even in ART 3.0 in Lisp, garbage collection is almost eliminated resulting in a significant increase in performance speed.

A CHOICE OF LANGUAGES

As expert systems continue to move into mainstream commercial computing environments, it is clear that users will want to develop and deploy expert systems in a variety of languages. Lisp, the traditional AI language, will remain the preferred language for exploratory development of software systems and product. In commercial environments, users are demanding expert system applications that will run fast and that can be interfaced to existing software in traditional languages such as C. In the longer term, ART 3.0 will be delivered in a variety of languages.

ART 3.0 in C and Lisp are the first steps in offering expert system developers a choice of development and delivery languages.

PRICING AND AVAILABILTY

ART 3.0 is available immediately for Symbolics^{TM}, Lisp Machines Inc. Explorer^{TM}, DEC^R VAX^{TM} (under VAX-Lisp) machines and the Sun-3^{TM} (under Sun Common Lisp) workstation. The C port for the VAX, the Sun-3, Apollo and IBM^R RT Personal Computer^{TM} will be available late in calendar 1986. Pricing is $65,000 for the first copy and $45,000 for copies two through five. Current ART 2.0 users having active maintenance contracts will receive

Inference

ART 3.0 as an upgrade. All versions include ART's graphics capabilities, Studio^{TM} and ARTIST^{TM}.

Inference supplies commercial expert system development tools, consulting and training to the aerospace, financial and manufacturing industries. Inference's ART provides AI programmers with a general-purpose industrial class tool for developing significant expert systems applications.

\# \# \#

Automated Reasoning Tool, ART, Studio and ARTIST are trademarks of Inference Corporation.

IBM is a registered trademark of International Business Machines Corporation.

RT Personal Computer is a trademark of International Business Machines Corporation.

VAX is a trademark of Digital Equipment Corporation.

DEC is a registered trademark of Digital Equipment Corporation.

Sun-3 are trademarks of Sun Microsystems Inc.

Explorer is a trademark of Texas Instruments, Inc.

Symbolics is a trademark of Symbolics, Inc.

Inference

A - 10

ELECTRONIC DESIGN EXCLUSIVE

Tool picks up pace of expert systems

*Practically abandoning 'garbage collection,'
an expert-system shell hastens processing,
aided by a quick pattern-matching scheme.*

BY MAX SCHINDLER

Most Lisp-based expert systems have trouble with real-time applications. For one thing, checking hundreds or thousands of rules at every decision point takes time. More subtly, the underlying Lisp language devours memory, for the programmer does not assign storage. Rather, during execution data spreads through memory haphazardly.

To reclaim storage space that is no longer needed, a "garbage collector" must determine which locations have become disconnected from the program flow. Some systems stop execution for as long as several minutes to let the garbage collector pass through. A "napping" CPU obviously cannot process data.

The first system to overcome the handicap is ART 3.0, the latest version of Inference Corp.'s automated reasoning tool. The software, essentially an expert-system shell, produces systems that are 99% garbage-free. Equally impressive, it draws on an unusual practice for joining rules in its knowledge base, a practice that speeds operation also. Finally, it recasts object-oriented programming with a brand-new way of associating objects.

ART 3.0's garbage-free operation grows out of a very basic concept. Inference designers invoked no magic: They simply did what programmers in other languages were doing for some time—explicitly deallocating storage when it is no longer needed. With no time spent collecting garbage, the system can trim workload by two orders of magnitude compared with its predecessor, ART 2.0. In the bargain it consumes 10 times less storage.

Speed benefits also derive from a pattern-matching technique that marks the first use of a novel rule compiler. A generalized topology joins rules in the data base from right to left, as well as from left to right as is customary.

Consider a rule that involves a great deal of matching—or "joining"—operations. The premise is this: If motor X and motor Y both consume 1 kW, and motor X produces more power than motor Y, and motor X is lighter than motor Y,

then motor X is preferable.

Now suppose that the data base contains 100 motors, and in only one case does a lightweight motor produce more power than heavier units. Ordinarily, determining whether motor 6 and motor 77 are both ac motors takes 10,000 comparison (100×100). The comparison at weight 77 dictates another 5050 operations (the sum of $1 + 2 \ldots + 100$). All told, the system must store 15,251 comparisons, including 201 comparisons for other weights and powers. Computation time is proportional to that total (see the figure).

ART 3.0 reduces the comparison tally to 401—38 times fewer—because the join tree hierarchy can operate from both the left and the right. But since matchmaking is not an expert system's sole occupation, the overall speedup can be diluted.

Traditionally, object-oriented programming systems associate a permissible procedure—a "method" —only with an individual object that is part of a hierarchy. Methods pass downward, with specific objects inheriting procedures from general objects above them. In the
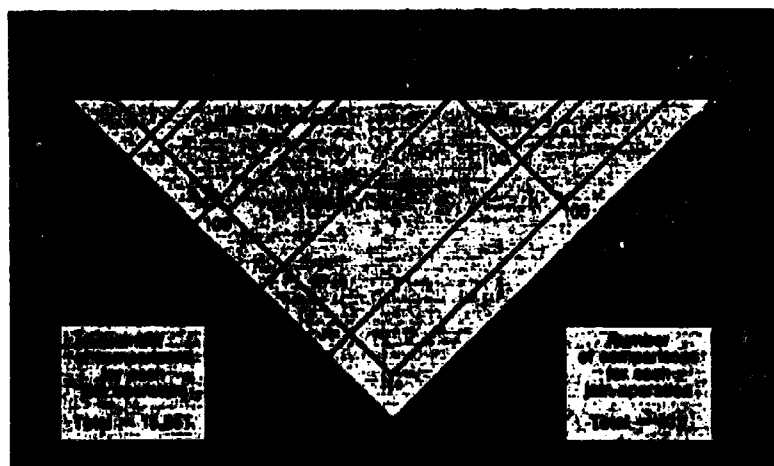
Conventional expert systems match rules from only one direction. For instance, in a comparison of electrical motors, a system would match—that is, "join"—facts only from the left (black). By breaking down the tree, Inference's ART 3.0 expert-system shell reduces the number of join operations nearly fortyfold (color).

absence of a specific method, default reasoning allows general methods to be inherited from those levels at which they are specified. Thus, a program can still respond.

The association and inheritance of methods, however, is suited only to a small number of programming paradigms, those in which a "tree" of operations or actions can be associated with a single object. ART 3.0 extends the hierarchical concept through a unique form of deterministic computation called multimethods. The technique attaches methods both to individual objects and to object types called vectors, which comprise groups of objects. Multimethods affords yet another degree of flexibility, because the knowledge base can be modified at run time without additional rules.

An object-oriented system also simplifies interfaces. For example, a programmer using multimethods can define a function that prints any part of a document on any particular device. The program is easier to write with an object-oriented system, while the rule-based capability lets programmers add more devices without rewriting the original code.

By combining object-oriented programming with multimethods, procedures specified for each class of printers can be associated with collections of documents. The technique thus covers a much wider class of problems.

Another new feature speeds up the software's execution still more. Called a file compiler, it converts the knowledge base into a conventional binary object file. Other expert systems compile their rules during loading, which slows the loading time for the end user, about tenfold.

Although shells like ART accelerate the development of an expert

system tremendously, they traditionally recompile after every change. Thus, development speed benefits greatly from the file compiler. ART-3.0's ability to run directly on binary code yields another side benefit: The system developer can keep the rules proprietary. □

Inference

Editor: Paul Harmon   Managing Editor: Michael Larrance

Reprint   July 1986

Article appears in its entirety,
as it appeared in the July 1986 issue of
EXPERT SYSTEMS STRATEGIES

Excerpted for Inference Corporation of Los Angeles, CA.

TOOLS

LARGE HYBRID TOOLS: ART, KEE, AND KNOWLEDGE CRAFT

OVERVIEW

In the last two years, three commercial
expert systems-building tools have
emerged as the "top-of-the-line" options
for R & D groups that want to explore the
development of large expert systems.
Some would argue that these three
software packages should not be called
"tools," but "tool kits" or "knowledge
engineering environments" to reflect the
fact that they each offer a variety of
different ways to approach any given
problem. We prefer to call these
packages large, hybrid tools, but we
certainly agree that they are
considerably more complex and offer more
options than any of the large rule-based
tools, such as S.1, E.S. Environment/VM,
or Knowledge Workbench.

THE THREE LARGE HYBRID TOOLS

ART is sold by Inference Corporation.
It was first introduced in March 1985.
The current version of ART, Version 2.0,
was introduced in January 1986.
Approximately 350 have been sold.

KEE was introduced in August 1983 and is
sold by IntelliCorp. The current version
of KEE is Version 3.0, which was intro-
duced in August 1985. Approximately 600
copies of KEE have been sold.

Knowledge Craft is a product of Carnegie
Group. It was first introduced in April
1985. The current version is 3.0 and a
new version 3.1, which will combine
Knowledge Craft and a natural language
package, Language Craft, will be released
at AAAI next month. Approximately 130
copies of Knowledge Craft have now been
installed.

APPROACH

We were not able to personally develop
and test a common problem on each of the
three tools being reviewed. Instead, we
talked with vendors, reviewed literature,
and watched demonstrations. In addition,
we studied reviews of these three tools
that were done by corporations seeking to
evaluate the tools, and we interviewed
some of the individuals involved in those
studies. We also talked with researchers

from a number of companies that have
purchased one or more of these tools.

We will compare and contrast the three
tools according to seven broad criteria:

1) Overall power and flexibility,
2) The knowledge engineering (or
   developer) interface,
3) The user (or runtime) interface,
4) The systems interface,
5) The runtime speed,
6) Training and support, and
7) Cost.

1. Overall Power and Flexibility

All three tools are written in Lisp and the
developer is always free to drop into Lisp
to write any additional code that he or she
requires for a particular problem. It
might appear that anything you can do in
one of these tools you can do in the
others, if you just take time to program
some special utilities. We don't disagree,
although each tool makes it much harder to
do certain things than others. For this
review, however, we will only discuss
features and utilities that are documented
and are compatible with the overall archi-
tecture of the particular tool. If we did
not establish this boundary on the features
that can reasonably be considered integral
to parts of each of these tools, we would
end up simply comparing Lisp to Lisp,
ignoring the motivation that would lead
someone to buy a tool in the first place.

ART

In essence, ART is a forward chaining
rule-based system derived from OPS5.
A great deal has been added to this
essential framework, but there is a strong
sense in which, if you use ART, you
approach problems from a rule-based
perspective.

Representing Knowledge

ART has four major components: facts,
schemata, rules, and viewpoints (or
contexts). The developer stores declar-
ative knowledge as facts, schemata, or
contexts while procedural knowledge is
encoded in rules.

The primary way of conceptualizing know-
ledge in ART is to think in terms of facts
and rules. Schemata, in ART, serve
primarily as "macro" forms in which to
express facts. That is, the developer can
use schemata to create object-relationship
semantic nets conceptually, but procedur-
ally, ART compiles it all into facts and
rules. One nice feature of ART's schemata
system is that it allows you to define
relations between schemata, which in turn
enables you to traverse a path through the
database in any direction.

ART supports two types of rules: state-
based rules that are, in effect "If...Then"
rules, and logical rules that take the
form: "While...Then," and thus establish
facts as long as other conditions are true.
All rules can be assigned salience to
determine the priority of their firing.
ART's rules permit Lisp calls from either
an If-clause or a Then-clause.

Since knowledge is kept primarily in rules,
as the size of the knowledge base
increases, maintenance becomes more
difficult than it would be with a
frame-based system.

Inheritance

To implement hierarchies of relationships,
the developer must create a substantial
rule set. The inheritance relationships
the developer can use are predetermined in
ART. This is necessary to allow ART to
pre-compile the knowledge base into working
memory before inferencing can begin, which,
in turn, allows ART its impressive runtime
performance. Still, the result is that ART
does not provide truly dynamic inheritance
on slot values in schemata.

Alternative Worlds or Viewpoints

ART's viewpoints provide a very effective
form of hypothetical reasoning. In effect,
each viewpoint is a separate scenario. In
this manner, the system can simultaneously
consider several scenarios, dropping them
as they become inadequate. ART can reason
about a viewpoint in the same way that it
can reason about facts. ART's viewpoints
can be especially valuable when reasoning
about events in time.

Figure 7. Screen showing an ART application

Truth Maintenance

ART provides a very useful logical dependency facility. If the If-clause of a rule is satisfied (and a logical dependency clause has been associated with that clause) the system will keep track of any subsequent inferencing that follows from that clause. If the clause subsequently becomes invalidated, any assumptions made on the basis of that clause will be automatically retracted.

Inference and Control

ART provides full forward and backward chaining capabilities. The primary control mechanism in ART is a version of the blackboard architecture.

KEE

In essence, KEE is an object-oriented programming system derived from Units, an expert system-building tool originally developed at Stanford University. Though much has been added, when using KEE, the developer initially approaches a problem by identifying the objects in the problem domain. KEE takes full advantage of its origins in Interlisp and object-oriented programming to provide elaborate windows, icons, and displays of objects and their relationships.

Representing Knowledge

The primary knowledge representation paradigm in KEE is the unit (or schema or frame). When using KEE, you begin by conceptualizing a problem in terms of objects and the relationships between objects. KEE uses graphics to provide the developer with a graphic overview of the objects and their relationships as they are developed or modified. The developer can easily begin to analyse a subject-matter domain without having to consider rules or procedures. For problems that involve a large descriptive or structural component, this is a very important advance over rule-based approaches.

Figure 8. Screen showing a KEE application

Rules in KEE are subordinate objects attached to the slots of higher-level objects. Demons can also be attached to slots in KEE. They are, in effect, rules that respond whenever the slot value is accessed or changed.

KEE has multiple rule classes which restrict the search space for rule firings. The rule class for execution can be selected dynamically. Multiple knowledge bases can be accessed in KEE, so knowledge can be passed from one knowledge base to another.

The KEE rule editor is very slow when the rules become more complex, and more sophisticated customers thus tend to rely more on Lisp for the control and program actions rather than on the rule language.

Inheritance

KEE provides an inheritance system with system-defined relationships such as "member" and "subclass". The developer cannot modify the inheritance system. This is a significant disadvantage since there are several types of relationships

that cannot be successfully conceptualized in terms of member/subclass relations.

The developer can specify restrictions on slot values, but only with a limited number of developer-set flags. The developer cannot write Lisp methods to handle exceptions (e.g., to query the user for specific information). This slot restriction mechanism poses real problems when one is trying to develop systems in which constraint propagation plays a large role (e.g., systems to handle design and configuration problems).

KEE can only support changing facts. It cannot support changing relationships; if a relationship exists in one context, it must exist in all. Thus, in KEE, all dynamic information must be stored as slot values rather than as explicit relationships.

Sustained use of object-oriented programming, including methods and message passing, allows the developer to create quickly a highly modularized system.

Alternative Worlds or Contexts

KEE can support only a very limited form of data changes in contexts and cannot
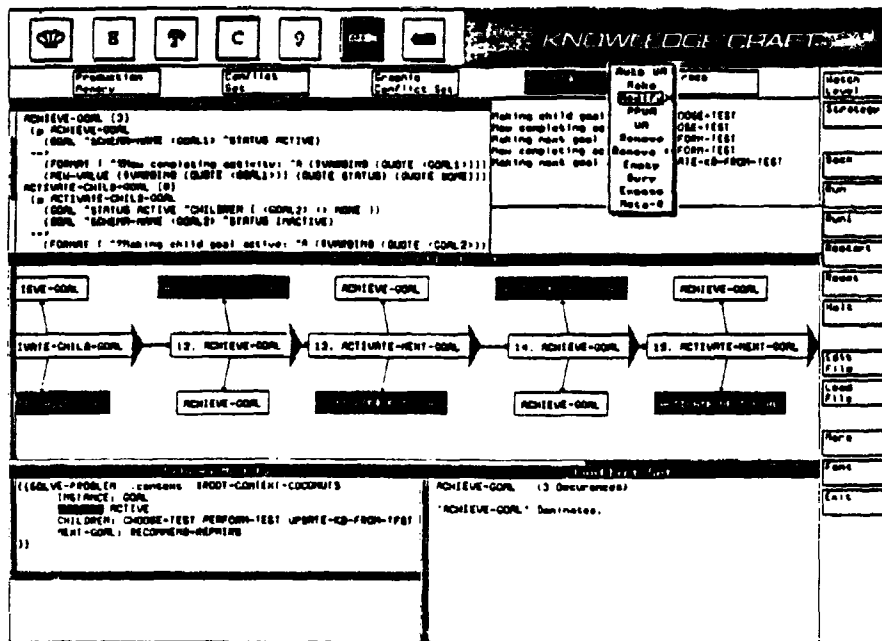
Figure 9. Screen showing a Knowledge Craft Application

support changes to system relations in contexts at all.

Truth Maintenance

KEE does not support truth maintenance.

Inference and Control

KEE supports full forward and backward chaining. However, KEE's Prolog lacks "cut" and "fail" and "cannot prune" and hence the developer cannot limit search in an efficient manner.

KNOWLEDGE CRAFT

Knowledge Craft is based on a semantic net approach derived from SRL, a schema or frame-based paradigm originally developed at Carnegie-Mellon University. More important, however, it is a collection of several more or less independent paradigms.

Representing Knowledge

Knowledge Craft has three basic language components: OPS5, Prolog, and CRL. The basic knowledge representation paradigm in Knowledge Craft is the CRL schemata

network. Each schema can have any number of slots associated with it.

Knowledge Craft makes extensive use of the meta information associated with the slots in a schema to allow the developer to provide default values, demons, cardinality restrictions, and range and domain restrictions. Using the demon facit, for example, the developer can attach demons that frequently initiate the processing of significant events. In Knowledge Craft, the demon facit is itself a schema with slots for the types of slot access that will trigger the demon, and when and what action the demon will take. Thus, demons become a fundamental part of the representation. In addition to the meta facits automatically associated with each slot, the developer can add additional facits when needed.

Knowledge Craft provides object-oriented programming techniques to permit data abstraction, object specialization and the passing of information via messages.

Inheritance

In Knowledge Craft, inheritance is specified at the relation level. This means

that the developer can specify which slots
and values can be included and which can
be excluded in any particular relationship
at the time that the relationship is
established. Thus, as you establish a
relationship, you can specify: hierarchies
of relations, the transitivity of
relations, and the semantics of the
relation, including the inclusion and
exclusion of slots and values during
inheritance (which slots/values can be
inherited), the mapping of slots/values
during inheritance (slot names or values
can change), and the elaboration of slots
(one slot can become many when
inherited). Thus, while the other tools
tend to force you to use hierarchical
relationships (e.g., ISA), Knowledge Craft
supports relationships like "sometimes-
leads-to" and "has-repair." Knowledge
Craft also provides the developer with the
ability to specify a search path to reduce
search time.

Alternative Worlds or Contexts

Knowledge Craft provides a context mechan-
ism which allows for different versions of
the knowledge base. This is used to model
and test alternative situations. Both
facts and relations are represented in
schemata. Contexts can be created and
schemata placed in them. These schemata
can be completely new or modified from
older schemata. Thus, Knowledge Craft
provides the option to arbitrarily change
any part of the knowledge base in any
context. In other words, Knowledge Craft
allows the creation of alternate worlds by
duplicating schema with hypothetical
information while retaining the original
copy.

Truth Maintenance

Knowledge Craft lacks truth maintenance.

Inference and Control

Knowledge Craft provides forward chaining
primarily through OPS5 and backward
chaining via Prolog. An agenda control
mechanism is also provided to allow the
developer to control multiple knowledge
sources.

2. The Knowledge Engineering Interface

All three tools provide some utilities for
developing menus and windows, for creating
graphics, and for controlling how end-
users will interact with the system.

To use ART, the knowledge engineer must
typically employ either the ZMACS editor
or the ART Studio, which provides utilities
that can be used to browse the knowledge
base, examine facts and rules, etc. In
addition, an ART Imagery Synthesis Tool
lets the developer create windows, menus,
icons, etc.

KEE clearly offers the best knowledge
engineering interface. The developer can
easily browse through the objects and
their relationships and develop graphic
icons from a large collection. It is easy
to use KEE's "active images" utility to
associate icons with demons related to
slot values and hence to assure that the
image on the screen "echos" a change in
the slot value. Using KEE, it is easy to
rapidly prototype a system. It is also
easy to maintain a system using KEE.

With KEE, the developer has the ability to
break forward or backward chaining in order
to assert or retract facts during a run.
KEE also provides many ways to call Lisp
functions. Unfortunately, many of these
calls are undocumented and there are no
advanced manuals to provide help. There
are also a number of "switches" that change
the way the rules execute, but these are
likewise difficult to discover.

The KEE rule display editor is very slow
when the rules become complex. Many
knowledge engineers use the ZMACS editor
once they are familiar with KEE.

To develop a system in Knowledge Craft,
the knowledge engineer can use either the
Knowledge Craft "workbench" or the ZMACS
editor. The components of the tool are
not well integrated and hence the develop-
er has difficulty getting started and must
program some routine connections to develop
a prototype. In addition, Knowledge Craft
relies on some relatively old technology,
such as its PALM network editor and its

schema editor, that are not as easy to use as those provided in ART and KEE. In effect, <u>Knowledge Craft requires more thought before you can get started, though its powerful environment may provide an adequate payback for more-sophisticated developers once they have made this initial investment and wish to extend or deepen their system.</u> Carnegie Group claims that version 3.1, to be introduced at AAAI next month, will substantially improve both the integration of their tool and the knowledge engineering interface.

3. The User (or Runtime) Interface

All three systems allow the developer to tailor the user interface in any way. None of the systems provide a pre-formatted interface that can be used by default, as the simpler systems do, but most users of these tools will probably prefer to develop their own interfaces. Of the three, KEE makes it easiest to develop a user interface, ART is helpful, and Knowledge Craft provides only limited help.

KEE makes it very easy to use graphics and images associated with active values. All of the vendors apparently assume that developers will either want to allow the user to access the underlying knowledge base or will be willing to program a runtime version of the system.

Rules in both ART and KEE are written in a natural language format that facilitates the development of an English-language explanation facility. Carnegie Group sells a special tool, Language Craft, that can be used to develop a more sophisticated natural-language interface.

4. Systems Interface

ART was written in ZetaLisp and is current- ly compatible with Common Lisp. ART is available on the following hardware: Symbolics, LMI, TI's Explorer, DEC VAX, and DEC MicroVAX. Inference Corp. has announced that ART will become available on the IBM PC RT during 1986.

KEE was written in InterLisp-D and is currently compatible with Common Lisp. KEE is available on the following

hardware: Symbolics, LMI, Xerox 1100, TI's Explorer, Apollo, Sun, DEC VAX, and DEC MicroVAX. IntelliCorp has announced that KEE will become available on the IBM PC RT during 1986.

<u>KEE provides customers with the possibility of using either PC or Macintosh terminals, connected to a larger system, as delivery terminals.</u>

Knowledge Craft is written in Common Lisp. It is available on the following hardware: Symbolics, TI's Explorer, DEC VAX, and DEC MicroVAX. The interface between the tool and the hardware, despite considerable effort, is still rather awkward. Carnegie Group has announced that Knowledge Craft will become available on the HP AI work- station and on the IBM PC RT during 1986.

ART and Knowledge Craft allow developers to access other languages such as C and Pascal that are supported on the hardware on which they are operating.

5. Runtime Speed

<u>Of the three tools, ART clearly runs fastest,</u> though it hardly runs as fast as an application written in a conventional language. ART achieves this speed by reducing its knowledge base to a sequence of facts and then compiling them, using the Rete algorithm developed at Carnegie- Mellon.

Both KEE and Knowledge Craft can take quite a bit of time to run, especially if they need to process a large number of rules.

6. Training and Support

All three vendors offer courses to provide customers with training and consulting support to help customers with specific applications.

Of the three, <u>Inference Corp. offers the best documentation.</u> It's not only compre- hensive, but it is skillfully written, flows very smoothly, and uses excellent examples. IntelliCorp and Carnegie Group each offer adequate but less comprehensive documentation written with less style.

Each of the vendors will work with a company to determine if their tool is appropriate for a particular problem. In some cases they will provide a tool for a test period.

7. Costs

ART costs $65,000 for the first copy. This price includes one year of maintenance and support along with free updates during that period.

KEE sells for $60,000. This price includes training and some on-site consulting.

The first copy of Knowledge Craft sells for $50,000. This price includes training for two people and maintenance, free updates and support for one year. Language Craft, the compatible natural-language package, costs $25,000 for the first copy.

All three vendors price multiple copies of their tools significantly below the price of single copies. There are also large discounts available to universities.

CHOOSING AMONG THE THREE TOOLS

To make a choice among the three tools, you really must to consider: (1) the nature of the problem, (2) the features offered by the tool, and (3) the skill and experience of your knowledge engineers. More than any other expert systems-building tools, the large hybrid tools require that the knowledge engineer know Lisp and have some skill in creating Lisp code to get the resulting system to perform.

1) If you have a specific problem . . .

If you have a specific problem in mind, then you will want to choose the tool that will best facilitate the development of a system to solve that problem, enable you to field the system, and allow for relatively easy maintenance of the system. As you consider your problem, you probably want to keep the following strengths and weaknesses of each tool in mind:

ART

Strengths

● If your problem can easily be conceptualized in terms of rules then ART will probably be the most natural tool to use.

● If you need to establish logical dependencies that will update your system dynamically as facts change, then ART's approach to truth maintenance and viewpoints will be very useful.

● If you will need to program special Lisp functions, the fact that Lisp functions are easily called from both sides of ART's rules will prove helpful.

● ART has the fastest execution time.

Weaknesses

● ART primarily keeps its knowledge in rules. This is convenient if the task can be conceptualized in terms of rules.

● Since ART relies heavily on rules, maintenance can become a serious problem as the number of rules increases, especially since all of the hierarchical structure in ART is also stored as rules. Significant maintenance problems begin when systems have over 2,000 rules.

● ART does not provide the best knowledge engineering interface and lacks good graphic editing facilities.

KEE

Strengths

● KEE has the best knowledge engineering environment, with superior graphic editing facilities, a large collection of icons, and good menus.

● Active values, including demons, methods, and active images, support data-directed reasoning and allow the system to recognize and monitor changing conditions.

● Object-oriented programming, including methods and message passing, allows convenient modularization of the expert system. In addition, KEE's multiple rule bases make it easy for the developer to partition the rule base efficiently.

Weaknesses

● KEE lacks a context mechanism and hence cannot easily represent multiple hypothetical situations.

● KEE has system-defined inheritance and will not allow the developer to tailor inheritance for special situations.

● The Prolog used in KEE is incomplete. It lacks "cut" and "fail," which makes it difficult to perform complicated reasoning using the rule interpreter.

● KEE limits the developers' access to a limited subset of Lisp.

● KEE lacks an agenda mechanism and thus the developer can not assert efficient control over the operation of the system.

KNOWLEDGE CRAFT

Strengths

● Knowledge Craft probably has the most powerful schema representation language of the three tools. It offers such features as dynamic inheritance, meta information, demon facilities, developer-defined relations, user-defined dependency relationships, and user-defined inheritance search patterns.

● The Context Mechanism allows the developer to create systems that entertain multiple hypotheses.

● The Agenda Mechanism allows the developer to tailor how the system will process a knowledge base.

● Object-oriented programming permits the conceptualization of problems in terms of objects, relationships, and messages.

● The implementation of Prolog has full resolution and includes "cut" and "fail."

● The senior staff at Carnegie Group has considerable experience in researching the design of large systems for factory scheduling and planning problems.

Weaknesses

● The integration of the various components is poor.

● The knowledge engineering interface is poor.

● The interface between the tool and the hardware is poor.

2) If You are Just Getting Started and Want to Explore . . .

If your AI group is new and you want to purchase one large hybrid tool to use to develop several initial prototype systems, you face a difficult choice. Each of these tools has significant strengths and weaknesses and each is clearly superior for some uses and not for others. We have summarized our overall impression of each tool in Table 1.

KEE has a really nice developer interface and will allow you to develop prototypes more rapidly than the other two tools. But it will fustrate you when you attempt more complex tasks, especially if those tasks have significant procedural elements and/or you are concerned with runtime speed.

Knowledge Craft is probably the most powerful and flexible of the three tools, but it has poor integration and a poor developer interface, which make it a hard tool to learn to use.

ART seems to be the tool that most companies are currently happiest with. Inference Corp. has struck the best balance between power and flexibility, the various interfaces, and runtime speed. ART's popularity may also reflect the fact that many developers are tackling projects that are more procedural than

1 = Poor
3 = Acceptable
5 = Excellent

	ART	KEE	Knowledge Craft
Power & Flexibility	4.5	4	5
Developer Interface	4	5	3
User Interface	5	5	4
Systems Interface	4	4	3
Runtime Speed	4	2	2
Training/Support	5	4.5	4.5

Table 1:
A Subjective Rating of the
Commercially Available Large
Hybrid Expert Systems-Building Tools

declarative and that ART's rule-based
approach seems more natural.

These comments have to be considered very
tentative. All three vendors are working
on new versions that will incorporate
updated features. Both Inference Corp.
and Carnegie Group will be demonstrating
new versions of their tools next month at
AAAI in Philadelphia.

OTHER OPTIONS

There are some other large hybrid tools
that we occasionally hear about. One is
Loops, the tool developed at Xerox PARC.
Loops is available from Xerox, but is not
really packaged for commercial use. (It
sells for $300 and only runs on Xerox 1100
machines.) Another large hybrid tool is
Schlumberger's STROBE which was developed
for internal use and is not currently for
sale.

Obviously, if you have a specific problem
that does not require the variety of
different knowledge engineering paradigms
provided by these tools, you should
consider one of the large rule-based tools
or one of the mid-sized tools. They cost
less and are generally easier to learn to
use.

On the other hand, if your problem is very
complex, these tools may not be up to it.
Organizations that have developed large
systems that involve real-time process con-
trol or reasoning about time have consis-
tently reported that they soon advance
beyond any of these three tools. At the
moment, however, if you want an expert
systems-building tool that will allow you
to tackle a large, complex problem, you
will want to consider these three tools.
They each have strengths and weaknesses,
but each represents a powerful way to
approach the commercial development of
large expert systems.

Company addresses are provided below. All
three companies maintain sales offices in
major cities throughout the U.S. and have
representatives overseas.

ART
Inference Corporation
5300 West Century Blvd., 5th. Floor
Los Angeles, CA 90045
(213)417-7997
Contact: Donald Gammon, VP Sales

KEE
IntelliCorp
1975 El Camino Real West
Mountain View, CA 94040-2216
(415)965-5633
Contact: Sue Brown

Knowledge Craft
Carnegie Group
650 Commerce Court, Station Square
Pittsburgh, PA 15219
(412)642-6900
Contact: Michael Chambers, V.P., Marketing
& Sales

* * * * *

IKE™

Integrated Knowledge Environment

Introducing IKE

IKE™ (Integrated Knowledge Environment) is a state-of-the-art, powerful, flexible, user-friendly, and complete environment for building expert systems and running goal-directed consultations. IKE is particularly well suited for rapid development of large expert systems that solve a variety of problems in the areas of diagnosis, configuration, decision support, and classification.

IKE has been designed so that people can easily and directly build sophisticated, functional expert systems without having prior exper-tise in knowledge engineering, the internal workings of expert sys-tems, or programming in LISP or any other language. This makes IKE the ideal expert system software for many professionals such as engineers, applied scientists, financial analysts, medical doctors, social scientists, business consultants, contractors, lawyers, educa-tors, operations and project managers, etc.

IKE provides natural language menu-based facilities for expert sys-tem development and use. These facilities are useful in:

- building and maintaining the domain vocabulary.
- writing and modifying inference rules.
- running interactive consultation sessions.

IKE uses extensive graphics to aid in expert system development, to trace reasoning paths, and to help manage knowledge bases. These aids not only greatly reduce the time needed for the development of an expert system, but they also help users gain a better understand-ing of both the nature of the problems they are trying to solve and the results of their IKE consultations.

Using IKE

IKE provides experts and consultation users with a menu-based natural language interface that enables them to concentrate on the task at hand: transfering knowledge into IKE and using it in consultations.

Each expert system that is developed in IKE has its own vocabulary. A medical diagnostic system, for example, will have one vocabulary and a diesel engine repair system will have another. A menu-based natural language interface is used both for the development of a domain vocabulary and in the construction of inference rules in terms of this vocabulary.

IKE's rule bases consist of if-then rules that encapsulate the expert's reasoning about the domain. Rules are written using both the domain vocabulary and a domain-independent vocabulary provided by IKE. As a rule is entered, IKE will, at every point, prompt the domain expert with all correct continuations of the rule.

IKE is a flexible environment, and is designed to be useful to professionals with varied backgrounds and areas of expertise. The full power of IKE is available through its high-level, structured natural language, graphical interface. In addition, for systems developers who need added programming functionality for their applications, IKE supports calls to LISP functions from within its operating environment.
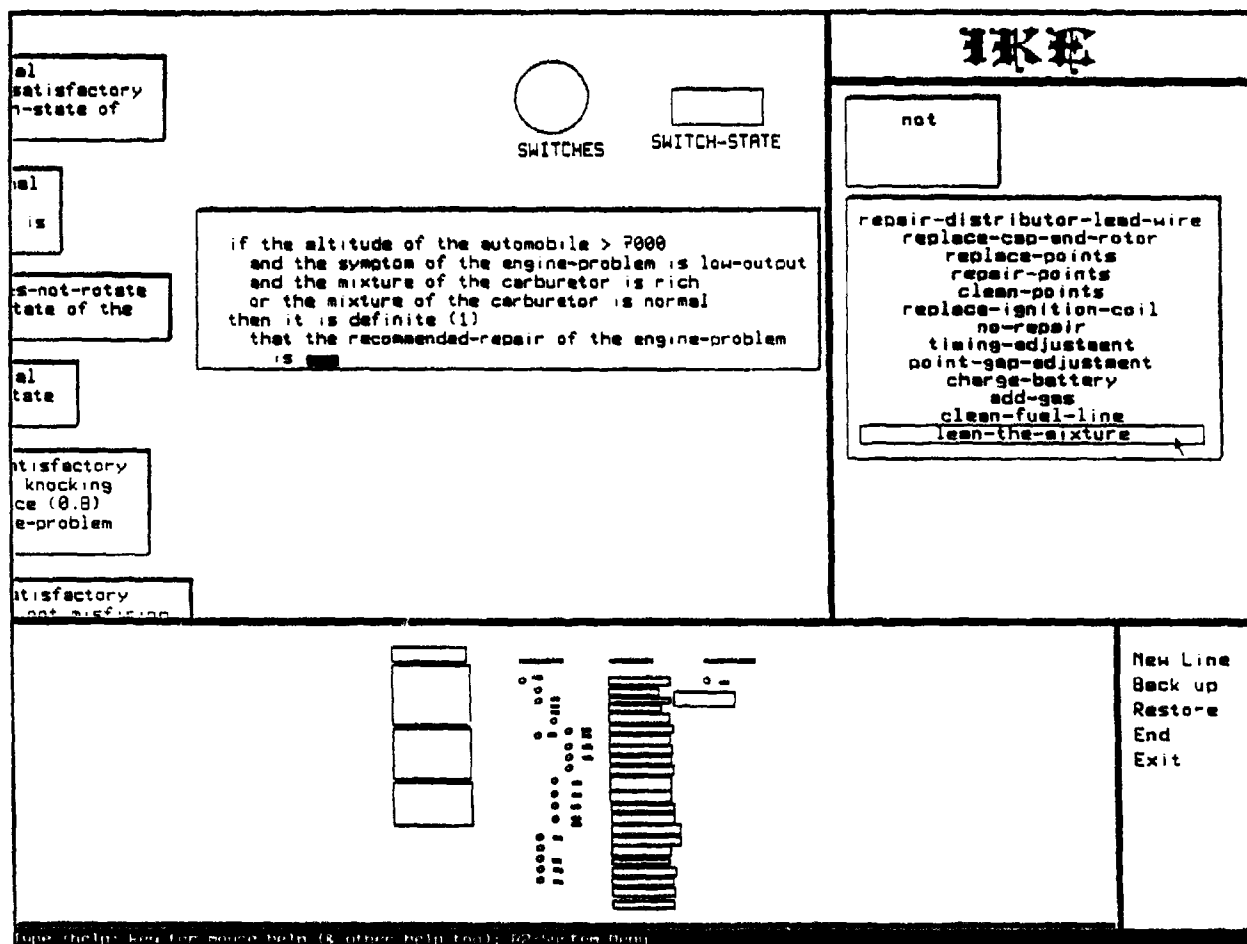
Illustration 1. IKE's domain knowledge is written and edited using structured natural language.

User Interface

The user interface is a complete and high level environment. With IKE, domain experts readily develop functional prototype systems within a week, and produce sophisticated applications within two to three months. IKE provides functionality at the right level of abstraction for its users. This greatly reduces start-up learning costs, and cuts development time by an order of magnitude.

IKE's user interface combines menu-based natural language and graphics to provide a flexible and natural consultation and development environment. The IKE user works within a structured form of natural language to rapidly and fully describe the domain, enter inference rules, and conduct consultations. In addition, IKE provides a menu-based interface to the knowledge base, allowing the domain expert to easily retrieve and modify rules and objects as development proceeds.

During consultations, IKE provides a graphical display of the developing inference tree and of the set of objects being reasoned about. This aids both the consultation user and the domain expert in understanding the reasoning that is taking place.

IKE's user interface guides users through system commands by providing step by step assistance. In addition, the user interface provides syntax checking and rule completion.

Reasoning

IKE's inference engine is especially designed for solving a variety of problems in the areas of diagnosis, configuration, decision support, and classification. The inference engine has both forward and backward chaining capabilities, and allows the user to volunteer information at any time before or during a consultation.

In general, the purpose of a consultation is to determine values for an attribute of an object based on information obtained from the user, from the knowledge base, and by inference. If a unique value of the desired attribute cannot be directly established, IKE reasons "backwards," looking for all rules in its rule base whose consequents may provide evidence about values for the attribute under exploration. Each rule that is applicable becomes part of the inference path. IKE must then find values of attributes used in the antecedents of these rules. To do this, IKE again searches the knowledge base for facts and rules that may provide evidence about values for these attributes. This process of alternation between seeking values for attributes and seeking facts and rules that provide evidence for establishing these values is how IKE does backward chaining. Backward chaining continues until either IKE establishes the needed values or, if it cannot establish these values independently, obtains them through querying the consultation user.

The domain expert can assign priorities to rules. These priorities are used to direct IKE to try certain inference paths before others.

If the user decides to volunteer information, IKE will forward chain to determine the implications of the new information for the ongoing consultation. Should the user change her/his mind about a previous fact, the user may input this change at any time during the consultation, and IKE will correct all inferences that need to be modified as a result of this change. This powerful feature is called truth maintenance.
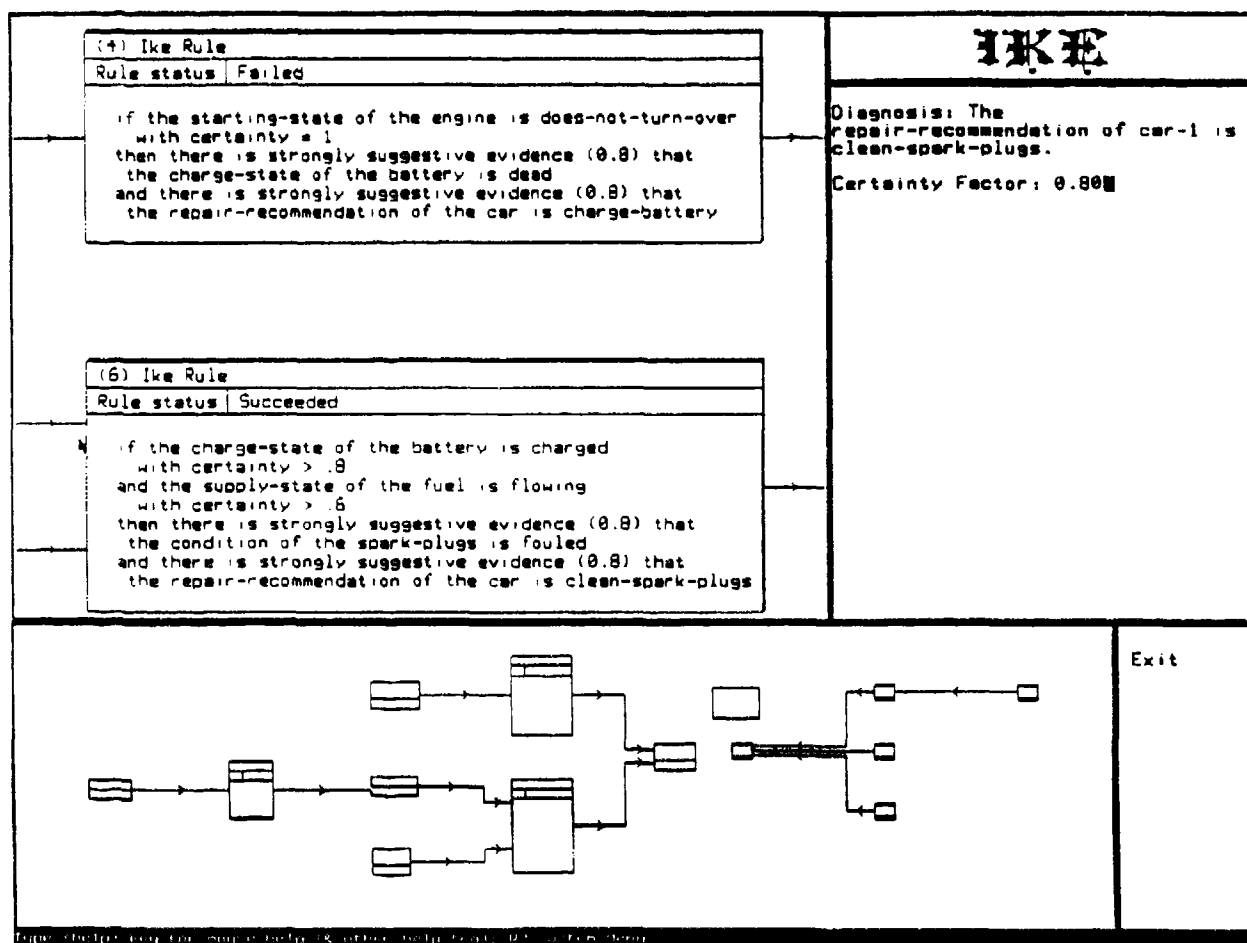
Illustration 2. During consultation, IKE builds a graphical inference history tree. The user can explore any part of this inference history and readily gain an understanding of how the consultation results were determined.

Uncertainty Management

IKE has facilities for applications containing elements of uncertainty, such as diagnosis of electronic devices, medicine, finance, and international affairs. IKE uses measures of confidence called certainty factors. Certainty factors are used to indicate:

- how strong or certain the association is between a rule's antecedents and its conclusions, or

- the accuracy, truthfulness, and/or reliability of the information provided by the user during a consultation.

For example, an economist may assign a confidence factor of 0.8 to an expected rise in the cost of world grain due to a poor long range weather forecast. The 0.8 indicates a strong belief, on the part of the domain expert, in the strength of the relation between weather and grain production. During a consultation, IKE may ask the user for information about a long range weather forecast. If the consultation user responds that drought is predicted with a certainty factor of 0.5, she or he is not challenging the domain expert's belief. Instead, the consultation user is expressing uncertainty about the definitiveness of the forecast.

Truth Maintenance

During a consultation, a user may want to volunteer information. Frequently, this information will change the values of earlier inputs, since

- new information may have come to light,
- the user may want to do what-if analysis or explore alternatives by changing certain key values, or
- the user may have developed a new opinion or need to change an incorrect value.

Changing values during a consultation affects the validity of inferences that have been derived as a consequence of the previous values. Many expert systems do not allow users to change values during a consultation. However, with IKE, when the user changes a value during a consultation, IKE's truth maintenance system checks and appropriately updates previous inferences that are affected by the changed value.

For example, in an automobile engine diagnosis, the fact that the headlights are working properly might lead to the inference that the battery is fully charged and that therefore, the engine does not turn over because the starter motor is broken. If the user later realizes that the headlights are dim (not working properly) and enters this new information, IKE will review the inferences based on this fact. With this new information, IKE might now infer that the battery is insufficiently charged, and then conclude that the engine does not turn over because the charge on the battery is insufficient for starting the car.

In some ways, IKE's truth maintenance is analogous to the recalculation capabilities of automated spreadsheets. In an automated spreadsheet, each time a value is changed, all calculations based on that value are automatically updated. In IKE, each time a value is changed, all *inferences and calculations* based on that value are automatically updated.

IKE's truth maintenance system provides great flexibility in consultation usage. Serious expert system consultations are often extensive, and the user will typically provide many values as the consultation proceeds. Without truth maintenance, changing just one of these values would cause the user to have to redo the entire consultation; thus, what-if analysis is infeasible. Truth maintenance makes IKE effective during consultations with changing data, and supports the full exploration of hypothetical alternatives.

Knowledge Base Management

IKE provides the user with extensive and powerful tools for knowledge base management. These tools allow large and sophisticated IKE applications, with thousands of rules, to be developed and managed.

The principle tools needed for the construction, updating, and modification of a knowledge base include searching and retrieval, replacement and sorting. In IKE, objects, attributes, and rules can be retrieved and sorted

- based on specific words or phrases,
- according to their author(s),
- according to their last date of modification, and
- based on specified types of problems (e.g. statements with undefined terms).

Once retrieved, these items can be easily edited and modified. Words and phrases in the IKE knowledge base can also be located and globally replaced.

In addition, IKE can perform retrievals on several conditions at once. which allows for tight specifications of retrievals from the knowledge base. Thus, domain experts can easily find subsets of rules focused on a specific topic where some contradiction or problem may have occurred. IKE's knowledge base management tools are the right set of tools to encourage and support joint and incremental development of large knowledge bases.

Modeling the Domain with IKE

Domain knowledge for each application is developed by the domain expert. In IKE, this domain knowledge is structured by:

- describing objects, including their attributes and default values;
- specifying basic relationships among these objects. These relationships include inheritance, roles, part/whole, etc.;
- specifying complex relationships among these objects using if-then rules (inference rules) with uncertainty factors;
- function definitions; and
- providing textual descriptions of specific objects (optional).

Knowledge of the domain is modeled in terms of objects, attributes, relationships, values, and rules. Objects in IKE are represented by "frames." In general, a frame consists of "slots" that describe different aspects or attributes of the given object. A frame-based representation allows objects to inherit attributes from higher level objects, and to have a variety of relationships to other domain objects. This makes it possible for the expert to develop the domain model in terms of higher level constructs. The domain expert can easily create or modify objects, attributes, object relationships and inference rules through the use of IKE's user-interface and graphics. Domain specifications and rules can be developed in IKE in any order.

Illustration 3. Objects and attributes in IKE are represented as icons. A frame corresponding to each icon can be selected for viewing and editing.

Portability and System Requirements

IKE is designed to run in ZetaLISP and CommonLISP environments. Knowledge bases created under IKE will be compatible across all implementations of IKE.

IKE currently runs on the LMI Lambda™, and the
TI Explorer™/LMI Lambda/E™ IKE requires 4 megabytes
of memory.

A CommonLISP version of IKE will soon be available for conventional workstations supporting CommonLISP including SUN-3™, DEC MicroVAXII™, and others.

Features and Benefits

IKE (Integrated Knowledge Environment) is a state-of-the-art, powerful, flexible, user-friendly, and complete environment for building large expert systems and running goal-directed consultations. IKE is particularly well suited for easy and rapid development of sophisticated expert systems applications. IKE is designed to be the expert system software that most professionals will want to use.

Ease of Use

- Complete high-level environment, with extensive graphical aids.
- Natural language menu-based interface.
- Full functionality of IKE available without any programming.
- User can build functional, sophisticated expert systems applications in weeks rather than months or years.

Powerful Inference Mechanisms

- Forward and backward chaining.
- Objects with inheritance.
- Uncertainty management.
- Truth maintenance (for exploring hypothetical alternatives).
- Monitoring and examining the inference tree.

The Right Kind of Knowledge Base for your Expert System

- Deep-knowledge modeling of objects and attributes.
- User enters knowledge in structured natural language.
- Extensive, graphics-based tools for creating, browsing through, and modifying the knowledge base.
- Default values for attributes.
- Facilities for documentation of domain knowledge.
- User may define and use mathematical functions.
- Supports joint and incremental applications developments.

IKE, LMI Lambda and LMI Lambda/E are trademarks of LISP Machine, Inc.
Explorer is a trademark of Texas Instruments.
Sun-3 is a trademark of Sun Microsystems, Inc.
MicroVAX II is a trademark of Digital Equipment Corporation.

Contact:
LMI
#4
6 Technology Drive
Andover, MA 01810
(617) 682-0500

IntelliCorp's

Knowledge Engineering Environment ™ (KEE) System:

Update

August 1986

This yeat at AAAI '86, the KEE system celebrates its third anniversary and third major release. This update provides a brief summary of the evolution of our customer base, KEE and IntelliCorp from 1983 to the present.

IntelliCorp's

Knowledge Engineering Environment ™
(KEE) System:
Update

1.0   IntelliCorp and KEE lead the industry.

IntelliCorp pioneered the commercialization of knowledge processing with the introduction of KEE in 1983. KEE is the most widely used, widely accepted software for building large, complex knowledge systems, with almost 950 copies of KEE licensed to over 250 sites, mostly Fortune 500 companies. KEE's functionality has evolved over the years, in response to our customers' needs, the market, and the advancement of the technology.

1.1.   The Evolving Family of KEE-based Products

With the introduction of SimKit™ in 1985, IntelliCorp was the first AI software company to offer 'second generation' development software that brings KEE's problem solving capabilities closer to the end user SimKit, built with and on KEE, represents a powerful marriage of knowledge system and simulation technologies.

1.2.   Economical Delivery of Completed Systems.

With the announcement of the PC-Host system in 1985, IntelliCorp offered the first economical capability for distributing completed applications to end-users on more traditional hardware without the time-consuming process of converting the application to conventional programming languages.

2.0   KEE-based applications

Today, KEE supports the development of diverse applications in a multitude of industries. Tomorrow, these applications are limited only by our imaginations. Customers are building a broad range of applications, including, but not limited to:

Alarm processing
Analysis
Data Interpretation/Data Fusion
Decision Support
Design
Diagnosis & Correction
Forecasting
Intelligent Advisor
Intelligent Database Interface
Layout & Configuration
Planning
Process control
Project Management
Real time diagnosis & control
Resource Allocation
Scheduling
Sensor interpretation
Simulation
Training

KEE customers represent the following industries:
Aerospace
Chemicals
Consultants
Electronics
Financial
Government
Insurance
Intelligence
Manufacturing
Military
Oil
Pharmaceuticals
Publishing
Telecommunications
Utilities

3.0 What is needed to build cost-effective knowledge systems?

As we work with our customers, our understanding deepens of what is required to build and deliver "industrial strength" knowledge systems. We believe that KEE 3.0 offers a comprehensive range of tools for appropriately addressing these requirements.

We have found that cost-effective knowledge systems usually have many different uses and users.. They are far more versatile and flexible than classic expert systems. They require an explicit and clear model of the problem domain which can then be accessed by various reasoning and analysis, and different users. (For further information on the requirements for building knowledge systems, see our INTELLINEWS on Model Based Reasoning.)

4.0 What is KEE?

You need more than reasoning tools to build knowledge systems. The hard problems center on acquiring representing, and interacting with *knowledge*. in a productive programming*environment*.. IntelliCorp has recognized this from the first. KEE's name, the Knowledge Engineering Environment is no accident. Br' ily, KEE is:

- Mature, proven, comprehensive software
- A comprehensive product -- integrating the best AI technology into a versatile range of tools
- Flexible, powerful, and productive environment for the full software cycle -- from prototype, through development to delivery
- Versatile --used for a wide range of large, complex knowledge-based systems, and by a wide bandwidth of users -- programmers (from the AI novice to the Ph.D. in AI), experts, and end-users (from top management to blue collar laborers)
- Easy to use and understand, accessing the power and flexibility of the all of KEE tools
- An open system to facilitate large application development, usability and extendability.

5.0. KEE 3: Functionality & Features

IntelliCorp's Knowledge Engineering Environment, which continues to evolve, is now in its third major release. With each release, IntelliCorp has extended KEE's functionality, while minimiziong the conversion process for its customers.

KEE offers advanced functionality for building models, reasoning about and analyzing those models, communicating with external systems --all in a clear, understandable and interactive manner. All of KEE's versatile tools are fully integrated. Not only is the integration of these tools a synergistic combination of features, but it is also the fundamental reason that KEE can support such a diverse array of knowledge systems so well.

5.1. KEE's REPRESENTATION LANGUAGE for clear, well-organized models.
Cost-effective knowledge systems requires a clear, well-organized, dynamic model. To this purpose, KEE provides the richest, most expressive, clear and flexible representation language available -- an *Object Oriented Frame-Based Representation Language*, enabling the representation of symbolic, autonomous objects in KEE. Each object knows all of the factual and behavioral information about itself. Users can send objects messages to elicit behavior, and objects can rapidly communicate with each other via message passing.

The basic building block for a model in KEE is the object, which is represented in a *frame* called a *unit*. Each unit is autonomous and self-knowing, containing all factual, behavioral and relational information about itself. The attributes of units are represented as *slots*. A unit can be thought of as an extended, sophisticated, very rich data record, containing not only static data fields, but also various *consistency checking* mechanisms, dynamic *subroutines*, plus *multiple inheritance* and *relations*.

The organization of large amounts of knowledge is crucial to successful knowledge systems. KEE's offers the most sophisticated tools for organization and information management. Objects can be organized into a *hierarchical relations* of class-subclass-member with full *multiple inheritance*. Classes of objects can be grouped together into modular *knowledge bases*. Knowledge bases can readily interact and communicate with one another since they share the common KEE language of frame representation.

KEE's representation language supports the building of a full, complex model. This fundamental model or "background world" can then be used for modeling *multiple worlds* with KEEworlds™. (KEEworlds is a new feature of KEE 3.0.) Some problems require exploring various alternatives simultaneously in order to rapidly arrive at a solution. KEEworlds provides powerful tools for exploring multiple hypothetical situations and alternatives. Consistency of knowledge is automatically maintained as worlds' states change through the most advanced *assumption-based truth maintenance system* (ATMS) available. The ATMS automatically keeps track of truth and contradiction, eliminates worlds that violate constraints, and provides explanation graphs to track system behavior. (Note: Johan deKleer, the originator of ATMS, is a member of IntelliCorp's Scientific Advisory Board.)

By integrating KEEworlds into KEE, IntelliCorp has advanced the state-of-the-art of AI development software. For the first time, a complete multiple worlds mechanism is integrated with a sophisticated object-oriented frame system. All of KEE's tools are available in all worlds, providing a powerful synergy in KEE.

5.2. REASONING & ANALYSIS TOOLS for problem-solving. The modularity of
KEE means that the model can be both physically and logically separated from the reasoning and analysis. This has profound implications for programmer productivity and software life cycle costs. KEE offers a variety of features for reasoning and analysis.

5.2.1. Production Rule System. KEE provides a powerful production rule system for efficient, controlled search including:

- Each rule is *represented as a unit* in KEE, and thus enjoys the full benefits of KEE's modeling language. Rules directly *reference and interact with other objects*, reasoning about the model. Rule premises and conclusions dynamically read from and write into slots value facets.
  Since rules are objects, they rules can be organized into *rule classes* for clarity and tasks

performance resulting from carefully directed and controlled search. Rules can belong to multiple rule classes and/or subclasses. This organization contributes significantly to programmer productivity and efficient control of search space.

- *Forward and backward chaining* with automatic *backtracking* when appropriate.
- Capabilities for *mixing chaining directions* in reasoning sequences, both in advance and dynamically at run time .
- *Clear, easy-to-understand* rule syntax.
- *User-controllable search* over a broad number of parameters for both forward and backward chaining is available and user-extendable. The search and conflict resolution strategies are both represented as slots in the rule class, so the value can be easily set in advance or dynamically at run time to prune the search space for optimization.
- *Deduction rules* to infer facts from other facts and state constraints, and to prune inconsistent, negligible and undesirable worlds. *Action rules* to add and delete facts in *existing worlds*. *Action rules* to spawn *new worlds* with specified facts.
- *Agenda mechanism* to control rule firing order, with functions provided for adding and removing from the agenda.
- Rules can take *actions*
- Logic operators available for referencing the frame structure in the *TellandAsk* ™ language. They extend beyond classic logic since they deal not only with True and False but also Unknown states. *Knowledge State Operators* evaluate states, and are limited to True/False. *Term Operators* can reference some part of a list of Values in the Value facet of a Unit.
- *Monotonic* and *non-monotonic* reasoning are both supported in KEE for solving a broad range of problems.
- Dynamic, highly interactive development, *debugging and explanation* facilities
- *Interpreter* for productive development and *rule compiler* for run time performance.
- *User-extensibility* of all parts of the rule system (i.e.,search strategies, parsers, agenda etc)

5.2.2. TellandAsk This logic language, with a clear syntax, provides facilities for building and modifying a knowledge base, retrieving information and initiating reasoning, and efficient direction and control of search.

5.2.3. Reasoning from Frames The deductive and retrieval capabilities of the frame system support direct and efficient reasoning from the frames -- often without invoking a rule. (See "The Role of Frame-Based Representation in Reasoning", Richard Fikes and Tom Kehler. *Communications of the ACM* , September 1985, Volume 28, Number 9.)

5.2.4. Reasoning across Worlds can be performed by the rule system or procedural Lisp code.

5.2.5. Object Oriented Programming with *methods*, Lisp procedures which perform conditional analysis and make changes to the system based on that analysis. Methods provide the fast, efficient message passing capabilities of Object Oriented Programming, and can be attached to any slot.

5.2.6. Data Directed Programming with dynamic *active values* or 'demons' provide a very fast mechanism for localized, direct reasoning and analysis when there is a state change. Active values are Lisp procedures attached to a slots that fire when the slot value is changed. They are 'smart', knowing how to monitor the slot value and respond to changes appropriately. An active value can be attached to any slot in KEE. In a rule-based systems, rules would have to provide this capability, adding significant overhead.

5.2.7. Access Oriented Programming with *activevalues* that fire when the slot value is retrieved.

5.3. INTERFACE TOOLS for ease of understanding. In order for a development environment to be truly productive, all of its powerful technology must be easy-to-use. This means being able to clearly understand it, and to interact with it simply and productively. Computer science has devoted much investigation into interface technology recognizing the significant role it plays in managing complex information systems. KEE integrates the most advanced and versatile interface tools available, providing a multitude of 'doors and windows' into KEE for clear and natural interaction. These support the full spectrum of users -- from the naive to the most advanced.

5.3.1. KEE's User Graphics Tools There is much truth in the old adage: "A picture is worth a thousand words." Cognitive science has shown that as much as 80% of human sensory input is visual. IntelliCorp has developed powerful graphics tools that take advantage of the graphics capabilities of advanced computer systems.

- *KEEpictures*™ help users construct customized, 'smart 'graphic images and interfaces that are machine independent and extensible. KEEpictures takes full advantage of object oriented programming, so each picture knows how to behave (i.e., draw, display, rotate itself.) This is a new feature in KEE 3.0.
-*ActiveImages*™ is a library of smart object images built with KEEpictures which can be attached to slots to *display or change* the current value of the slot. This dynamic, two-way communication is made possible by to object oriented programming. Images can be attached to slots iautomatically, and readily modified or customized. *Textual* images can be customized for natural interaction. Groups of images can be organized into control panels, which can be quickly shrunk and expanded as needed. ActiveImages have been enhanced for KEE 3.0.

5.3.2. KEE system interface tools are highly dynamic and interactive, offering various levels of interaction.
- *Dynamic multiple windowing* system that is easy to use. Multiple windows provide programmers with various views into the system simultaneously and are easily user-defined.
- *Desktops* Windows can be organized on a screen, with the information displayed as specified. A screen can be quickly identified as a desktop. The programmer can define multiple desktops and switch between them by simply selecting from a menu with a mouse. The end-user can be locked out from the underlying system for security through desktops. (This is a new feature of KEE 3.0.)
- *Mouse-menu* interaction in KEE provides productive and easy-to-use means for issuing commands to KEE. A variety of menus, with layers of sub-menus are available. And users can customize menus for development and delivery.
-*WorldBrowser* is a dynamic, active interface for exploration, interaction with and understanding of KEEworlds.
- *Rule debugging* is facilitated by the dynamic tracing of And- & Or -tree reasoning graphs. These graphs have active nodes for toggling breakpoint switches, and automatically calling up the rule under consideration. Toggle switches are also available for automatically turning on various debugging facilities such as stepper mode, and rule graphs and text traces.
- *Explanation graphs* of completed rule-based reasoning sequences are automatically available.
- *Dynamic editors* for building and editing knowledge bases units, rule classes, rules, etc.

5.4. EXTENSIBILITY, PERFORMANCE & COMPATABILITY.

5.4.1. Open architecture IntelliCorp chose to design KEE as an open system. This has enabled clear, rapidly accessible, modifiable and extensible KEE-based systems. Advanced programmers can easily customize and extend KEE in a number of ways because of the open architecture and synergistic integration of all of KEE's tools. This flexibility has allowed direct user input to KEE-based system in other programming languages (i.e., C, BASIC), and direct communication with external systems. Perhaps most important, it has allowed IntelliCorp to readily extend and evolve KEE to meet the needs of customers.

5.4.2. Fastest performance. KEE is optimized for speed at each stage of the software life cycle. During development, KEE runs automatically in *interpreted* mode, giving instant feedback to all changes. As such, it brings significant productivity gains. Once rules, ActiveValues, and Methods have been debugged, they can be *compiled* for quick execution. Other performance gains are rendered by the basic architecture of KEE. Object oriented programming brings organization and modularity, and the fastest data directed programming and information retrieval. *Rules classes* segment rules, efficiently controlling and directing search. *ActiveValues* and *Methods* localize reasoning and analysis for direct, rapid response.

5.4.3. Cross-machine compatability KEE is now available in *CommonLisp* with *CommonWindows*, making porting across different hardware virtually effortless. CommonWindows sit underneath all of KEE's graphics, enabling the porting of KEE-based applications readily across various hardware. The porting of full graphics is unique to KEE. Designed and implemented by IntelliCorp staff, CommonWindows is quickly becoming the industry standard.

6.0. Customer Services for Customer Success

6.1. Technology Transfer from Intellicorp to our Customers is achieved not only through providing state-of-the-art products, but through a strong range of services to support our customers in applying the technology to their business problems.

6.2. A wide range of responsive services are available to assist customers in putting KEE to work to meet their business goals. These services include:
- *Training* in the use of IntelliCorp's products. Hands-on approach tailored to each individual customer's problem. The KEE 3.0 training is a 9 day course. The last 2 days focus on building the customer's application.
- *Apprenticeships* -- An intensive one-month tutorial at IntelliCorp, the apprenticeship program teams a skilled IC Knowledge Engineer with the customer team to build a full prototype of the customers' application. This popular program provides a strong accelerator on the customer's learning curve.
- *Contract applications services* -- IntelliCorp has a large staff of experienced Knowledge Engineers whose services are tailored to meet the customer's needs, and span the range from very short contracts to assess the feasibility of building a specific application, to complete turnkey systems.
- *On-going technical support* -- IntelliCorp has a strong customer services team who work with KEE customers by telephone and on-site. Each customer has a customer services representative who provides continuity in the customer-IntelliCorp partnership. Based in California, the services telephone lines are open from 5 a.m. to 5 p.m. P.S.T.

6.3. Standard KEE Support Package IntelliCorp has found that the following set of support services provides an effective means of successfully launching and sustaining customers' use of KEE:

     9 days of training for a team of 2 people
     2 days of consulting (to be taken within 90 days of training)
     site support (on-going telephone support, software & documentation maintenance & updates)

Other services can be configured as appropriate to meet the customer's needs

7.0 Summary

This update has been intended to give a brief overview of IntelliCorp's Knowledge Engineering Environment release 3.0. IntelliCorp continues to invest heavily in research and development, integrating our advanced AI research into our current and future products. IntelliCorp will continue to evolve our current family of products, develop new advanced products, and engineering innovations for development and delivery of knowledge systems.

IntelliCorp is committed to continuing as the industry leader by anticipating the needs in the marketplace, and providing products & application support services to meet those needs.

IntelliCorp Regional Sales Offices

Atlanta, Georgia
404-980-6688

Cambridge, Massachusetts
617-868-5611

Chicago, Illinois
312-648-1060

Dallas, Texas
214-458-0737

Los Angeles, California
213-216-6944

McLean, Virginia
703-749-3790

Mountain View, California (IntelliCorp Headquarters)
415-965-5650

Munich, Germany
(011)49-89-41-43-65

New York City
212-418-0476

Philadelphia, Pennsylvania
703-668-1704
2/5

Donald H. Theune
Account Manager

IntelliCorp

IntelliCorp **Knowledge Systems Division**
Two Bala Plaza Suite 300
Bala Cynwyd. Pa. 19004
Telephone: (215) 668-1704
(703) 749-1431

KEE, KEEworlds, KEEpictures, ActiveImages, and SimKit are trademarks of IntelliCorp.

IntelliCorp is a registered trademark of IntelliCorp.

KES—AN EXPERT SYSTEM DEVELOPMENT TOOL

ABSTRACT

The Knowledge Engineering System (KES) is a tool that supports the rapid development of prototype or production expert systems. The system features three choices of knowledge representation and inference engine, and an English-like language for specifying the knowledge base of an expert system. An artificial intelligence (AI) or computer science background is not required to use KES to develop expert systems. This paper introduces the concepts of expert systems and knowledge engineering; describes KES; illustrates the expert system development process using KES; discusses some applications developed using KES; and outlines future product plans.

INTRODUCTION

Software A&E's Knowledge Engineering System (KES) is a set of tools for developing expert systems. KES can be used to develop expert systems in a wide variety of application areas found in commercial, industrial, government, and military environments. KES greatly simplifies the tasks of building, maintaining, and using expert systems. KES provides the reasoning and deduction, as well as the user interface, while the knowledge base author provides the 'knowledge' unique to the application.

ARTIFICIAL INTELLIGENCE AND KNOWLEDGE ENGINEERING

One of the main goals in the study of artificial intelligence (AI) is to create computer programs that perform actions which, when performed by humans, can be said to require intelligence [1]. More precisely, AI is the study of "the principles of intelligence using information processing concepts as its theoretical framework and the computer as its principal tool" [2].

Expert systems are computer-based software systems that achieve high levels of performance in task areas that, for people, require years of special education and training [3]. After many years in the laboratory, expert systems are now being put into operational use in a variety of applications. The applied sub-field of AI that deals with building expert systems is called knowledge engineering.

Conventional software systems can be thought of as applying an algorithm (the program) against a data base with a current set of data. In contrast, an expert system has the implementation of the control structure (the inference engine) separated from the 'rules' or other knowledge representation, which are stored in the knowledge base.

A partial list of current and planned applications of expert systems is

> medical diagnosis
> equipment failure diagnosis
> computer configuration
> chemical data interpretation/structure elucidation
> experiment planning
> speech and image understanding
> financial decision making
> signal interpretation
> mineral exploration
> military intelligence and planning
> advising about computer system use
> integrated circuit design

Some specific examples of expert systems and their developers are

DENDRAL	determines molecular structure from mass spectrometer data (Stanford)
MYCIN	diagnoses bacterial blood diseases (Stanford)
PROSPECTOR	advisor for mineral exploration (SRI)
DIPMETER	advisor for oil well drill sites (Schlumberger)
RI	configures computer systems (CMU/DEC)
STEAMER	trains machinist mates in shipboard repair and maintenance (Westinghouse)

A recent survey of the application of expert system technology identified the following advantages of its use as an alternative to conventional software architectures [4]:

- automation of tasks previously not feasible

- easier to use and understand

- spectacular increase in programming productivity

- genuine extension of human capabilities

THE KNOWLEDGE ENGINEERING SYSTEM

CHARACTERISTICS

KES eases the building and maintaining of expert systems. The knowledge base author does not need to be an AI or even computer expert. He uses a suite of KES development tools and the KES knowledge representation language, instead of LISP, PROLOG, PASCAL, or C. KES also eases the use of expert systems. The end user needs no specialized training. He accesses the knowledge base through a simple, but powerful, interactive interface.

KES is broadly applicable to many applications in a variety of environments. No specialized hardware is required. KES runs on the Digital Equipment VAX-11 under VMS or UNIX, the Apollo Domain, Sun, and Tektronix workstations, and the IBM PC. KES is domain independent which means it is useful in a wide variety of applications areas, and has multiple knowledge representations and inference engines. It also has facilities for semi-structured problems.

A typical interactive session of an expert system built with KES begins with a series of messages to the user explaining the purpose of the expert system and optionally giving instructions in its use. The session then continues with a sequence of questions. The user is asked to respond with his selection from a list of multiple choices, a numeric value, or an arbitrary string of characters (e.g., a name). During questioning the user can interrupt to ask for help, issue a KES command, or ask for a detailed explanation of the question. After the questions have been answered, the expert system provides its recommendation. The user can then ask KES for a justification of the result (i.e., the line of reasoning used to develop the recommendation).

ARCHITECTURE

Figure 1 illustrates the architecture of KES. The knowledge base author, who may be the expert himself or a knowledge engineer, uses the KES parser to create the operational knowledge base. The end user can access the knowledge base with the runtime system. Applications can call KES subroutines to access a knowledge base directly.
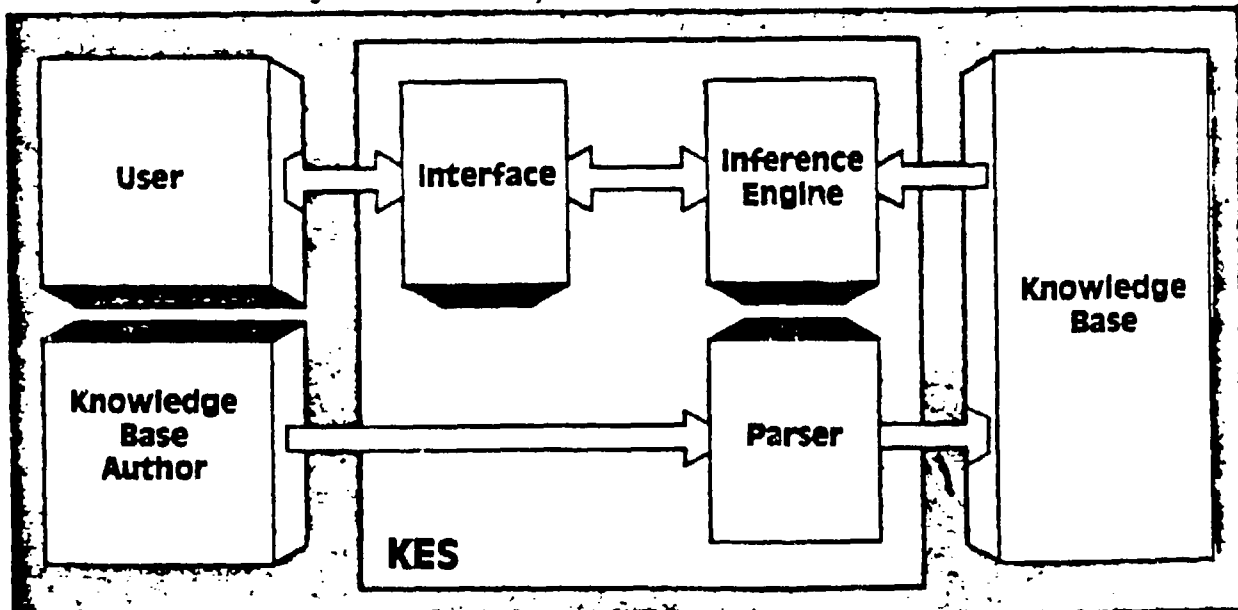

Figure 1: The Architecture of KES

FEATURES

KES supports three different knowledge representation and inference engine pairs: rule-based production system (PS), Bayesian statistics (BAYES), and hypothesize and test (HT). In addition, KES supports derivation of attribute values from calculations and invocation of external programs, as well as character string manipulation and pattern matching. The user interface is programmed with the procedural-oriented actions section of the knowledge base.

With PS, knowledge is represented by rules in the form IF antecedent THEN consequent. For example (with "|" standing for logical "OR" and "&" standing for logical "AND"):

```
IF NOISE OF WHEEL = SQUEAKING | GRINDING
    AND OIL = AVAILABLE OR GRAPHITE = AVAILABLE
THEN REPAIR = LUBRICATE THE WHEEL.
    MATERIAL USED = OIL < 0.8 > | GRAPHITE < 0.2 > .
    MESSAGE "PUT LUBRICANT ON THE WHEEL".
ENDIF.
```

is a rule which means if the wheel is squeaking or grinding lubricate it with oil or graphite. PS is goal- or consequent-driven with only the necessary questions generated to achieve a specified goal. Uncertainty can be explicitly represented with confidence factors as shown above for MATERIAL USED.

With HT knowledge is represented by frame-like descriptions. Each description consists of a collection of statements related to the phenomena of interest. For example, the following are two values from a knowledge base that diagnoses plumbing failures:

```
CAUSE OF PROBLEMS (MLT):
  WATER MAIN TURNED OFF OR PIPES FROZEN
    [DESCRIPTION:  NATURE OF PROBLEM = NO WATER SUPPLY],
  AIR CHAMBER FAILURE
    [DESCRIPTION:  NATURE OF PROBLEM = NOISE
      [LOCATION    = PIPE;
       NOISE TYPE = BANGING;
       OCCURRENCE = IMMEDIATELY AFTER WATER IS TURNED OFF].
```

The HT inference engine is based on the concept of minimal set covers to simulate an hypothesize-and-test approach to problem solving [5]. It determines the smallest number of causes, represented by descriptions in the knowledge base, that explain all known manifestations of the problem of interest.

With BAYES knowledge is represented by the prior probabilities of the
possible outcomes and the conditional probabilities of the determinants.
Expert systems built with BAYES are usually built when a statistical
experiment has been performed on a known population. Bayes Theorem is used by
BAYES to compute the conditional probabilities of the outcomes based on the
actual values of the determinants. For examples, the following is taken from
an experimental knowledge base for determining the prognosis for recovery of a
person who had a stroke:

    AGE: SIXTY AND ABOVE, LESS THAN SIXTY.

    TYPE OF STROKE: INTRACEREBRAL HAEMORRAHAGE,
                     THROMBOTIC INFARCTION,
                     EMBOLIC INFARCTION,
                     SUBARACHNOID HAEMORRHAGE.

    SEVERITY OF STROKE: MILD, MODERATE TO SEVERE.

    PROGNOSIS [DETERMINANTS: *]
      GOOD < 0.16 > 0.25 0.75;
                      0.10 0.20 0.40 0.30;
                      0.78 0.22,
      FAIR < 0.59 > 0.38 0.62;
                      0.15 0.25 0.41 0.12;
                      0.54 0.46,
      POOR < 0.25 > 0.82 0.18;
                      0.49 0.21 0.12 0.18;
                      0.19 0.81 .

The prior probabilities directly follow each of the value names, and the
conditional probabilities for each of the determinants are then listed in
order of definition and value.

APPLICATION SYSTEM ARCHITECTURES

    Two of the many expert systems that have been built using KES are the CDC
Dump Analysis Expert System and the Tactical Mission Planning Expert System.
The CDC Dump Analysis Expert System was jointly developed by Control Data and
Software A&E as a prototype to diagnose the cause of a Cyber NOS-VE system
crash. It first determines whether hardware or software caused the crash;
then which hardware module or what OS problem caused the crash, and, if
software, what project, project leader, and module was responsible. The
knowledge base contains over 140 attributes with many possible values, over
800 rules, and over 100 commands in the actions section. The demonstration
version took five days to develop and the current version about five weeks.

    The Tactical Mission Planning Expert System was developed by Software A&E
for the U.S. Army Engineering Topographic Laboratory to demonstrate the
concepts of expert systems applied to battlefield command and control. The
system deals with a hypothetical battlefield reconnaissance planning
situation. It determines what portions of the selected terrain may have
artillery batteries given the enemy's doctrine for deployment and the
characteristics of the terrain. This information would then be used to plan a
reconnaissance mission.

The system architecture is shown in Figure 2. The Tactical Planner and Supervisory Control portions are implemented as separate knowledge bases using PS. The Spatial Reasoner is an inference engine custom designed by Software A&E to deal with spatial relationships between physical objects. The Interaction System supports the user interface. The Graphics System supports the four color graphics displays used with the system. The Shared Information System coordinates access to the Symbolic Terrain Database. These last three systems were implemented with conventional software techniques.

KNOWLEDGE ENGINEERING WITH KES

A knowledge base author builds a KES expert system by developing a knowledge base. This process progresses in four steps much like the steps in developing a conventional software system.

```
 ┌──────────────────────────────────────┐
 │ TACTICAL PLANNING (KES.PS)            │
 ├──────────────────────────────────────┤
 │ SUPERVISORY CONTROL (KES.PS)          │
 └──────────────────────────────────────┘

 ┌──────────────────────┐        ┌INTERACTION SYSTEM
 │ SPATIAL REASONER      │        └GRAPHICS SYSTEM
 └──────────────────────┘

 ┌──────────────────────────────────────┐
 │ SHARED INFORMATION SYSTEM             │
 └──────────────────────────────────────┘

              SYMBOLIC
              TERRAIN
              DATABASE
```
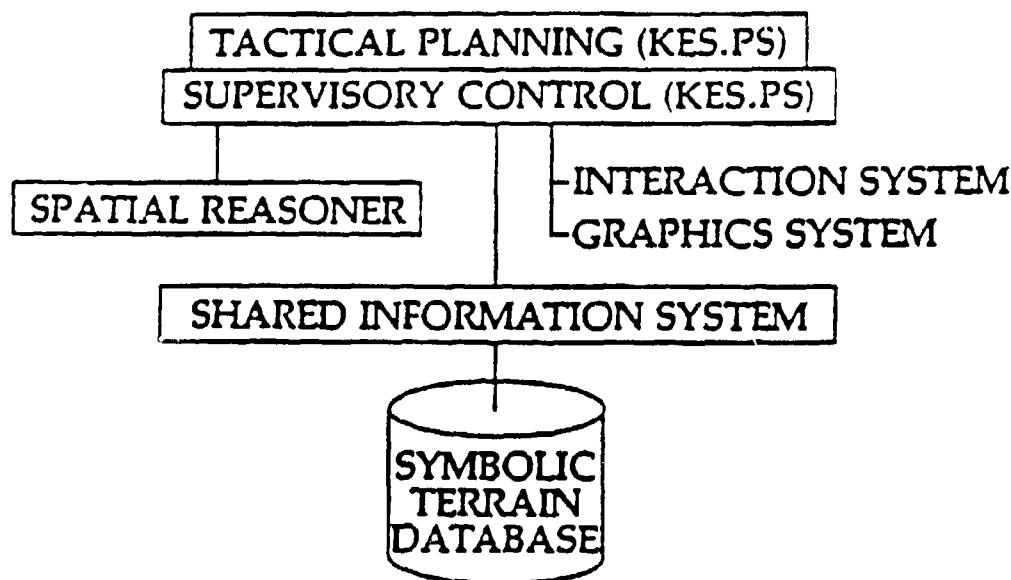
Figure 2. Tactical Mission Planning Expert System

The steps in the knowledge engineering process, along with their typical activities, are as follows:

> Analysis - the knowledge base author determines the scope of the knowledge required of the expert system by outlining the goals. He identifies the knowledge sources (experts, books, etc.)

> Design - the knowledge base author defines the attributes and their relationships. He chooses the knowledge representation and inference engine to use and sketches out what the end user interface will look like.

> Implementation - the knowledge base author encodes the knowledge base (attributes, rules, 'free text', user interface, etc.) in KES syntax and uses the KES Parser to create the operational form of the knowledge base.

> Test and Evaluation - the knowledge base author uses the KES Inspector to verify the knowledge base. He then performs static analyses and dynamic tests.

The knowledge engineering process is typically performed iteratively until the knowledge base author is satisfied with the performance of the expert system. Experience has show that expert system applications evolve from demonstrations (developed in days) through several prototypes (weeks to months) to the production system (months to years). The production version of the expert system will itself evolve over time as experience in its use is obtained.

SUPPORT

Software A&E offers full support for KES licences including telephone consulting, bug fixes, and updates. Software A&E also offers knowledge engineering and KES customization services. Training courses for KES are offered on a regular basis. Each course lasts one week and covers all aspects of expert system developement with KES. These courses are included in the price of most KES licenses.

The Knowledge Base Author's Reference Manual serves as a guide to expert system development with KES. This manual is designed to be useful to both experienced and novice expert system developers.

A seminar entitled "Building Expert Systems: An Assessment of State of the Art Artificial Intelligence Practical Applications" is offered periodically at various locations in the U.S. and Europe.

SUMMARY OF BENEFITS

The principal benefits of using expert systems technology are

- immediate availability of expertise

- elimination of individual bias, prejudice, and errors due to
  oversight or fatigue

- valuable teaching aids because of ability to justify conclusions

- automation of tasks previously not feasible

- capture of corporate knowledge

The principal benefits of using KES to develop expert systems are

- high level knowledge-representation language

- greatly reduced development time

- portability from personal computers to mainframes

- can be integrated with existing software

- suite of development tools

- simple, but powerful, programmable user interface

FUTURE PRODUCT PLAN

Software A&E's Next Generation Product (NGP) will be both a major step
forward in AI development tools and a fundamental advance in the manner of DP
application development. It will marry proven technologies from software
engineering, fourth generation systems, and fifth generation systems.

NGP will allow the user to:

- Develop stand-alone, real-time, and embedded applications in much
  less time and with substantially less drain on development resourc.s.

- Rapidly build prototypes of software applications.

- Develop requirements and production applications in parallel.

- Naturally combine symbolic and traditional computational techniques.

A diverse group of users, from DP novice (but domain expert) to experienced software/knowledge engineer, will be able to use NGP in their development efforts. NGP's unique architecture will allow these users to build an application to run identically in numerous different computer environments without change to the specifications.

This multi-million dollar product development effort is jointly funded with Control Data Corporation. Limited release of NGP is scheduled for 1986.

REFERENCES

[1] Nilsson, N.J. Principles of Artificial Intelligence Tioga (1983).

[2] Fox, J. "A short account of knowledge engineering" The Knowledge Engineering Review 1: 1 (November 1984).

[3] Hayes-Roth, F. et al. Building Expert Systems Addison Wesley (1983).

[4] Johnson, T. "The commercial application of expert system technology" The Knowledge Engineering Review 1: 1 (November 1984).

[5] Reggia, J.A., D.S. Nau, and P.Y. Wang "Diagnostic expert systems based on a set covering model" in Developments in Expert Systems (M.J. Coombs, ed.) Academic Press (1984).

Knowledge Craft™ Version 3.0
An Environment For Developing Knowledge-Based Systems

Overview

Reflecting years of research, design development, and extensive field testing, Knowledge Craft™ constitutes a highly integrated set of tools for constructing practical AI software solutions. By enhancing the productivity of experienced AI programmers, Knowledge Craft dramatically reduces the effort required to build knowledge-based systems.

The tools provided by Knowledge Craft facilitate a flexible approach to expert system design, and encourage rapid prototyping of applications software. The system offers programmers a choice of control strategies and knowledge representation techniques. Version 3.0 enhancements include OPS, Prolog, and CRL™ Work Centers, extended interface capabilities, and significant improvements to the window and command systems. Knowledge Craft features include:

- Frame-based knowledge representation language with inheritance and procedural attachment
- Database management
- Knowledge base editors
- Logic programming
- Rule-based programming
- Object-oriented programming
- [...] management
- Programming workbench
- Device independent interface tools with 2D graphics and icons
- Complete Common LISP integration

Knowledge Craft's approach is based upon proven experimental prototypes developed at Carnegie Mellon University and the experience of solving large production problems in a broad range of industrial environments.

Applications

Knowledge Craft is now being used in many application areas:

- Project management
- Factory planning and scheduling
- Product selection
- Alloy design
- Computer aided design
- Process diagnosis
- [...]

CRL™: Carnegie Representation Language

CRL™ provides a frame-based language that is efficient, easy to use and suitable for both large and small applications. The basic unit of representation is the *frame* or *schema*. Each schema has slots and values to store attributive and relational information about an entity. Relations link schemata to one another. Information in one schema can be transferred to another using simple taxonomic inheritance, or more sophisticated methods.

- **User-defined relations and inheritance semantics.** In order to facilitate world modeling, users may define new relations. Real world relations, such as *part of*, *version of* and *required for* may be created. For each relation, the user can specify inheritance semantics, i.e. which slots and values are to be inherited over that relation.

- **Meta-knowledge representation.** Knowledge about the representation itself can be systematically partitioned from knowledge about the domain being modelled. This facility is used for chronicles, knowledge base development, maintain dependency information, and provide local specialization. The tailoring of relations and schemata by schema basis.

- **Dependency representation.** Knowledge Craft maintains dependency relations for inherited slots and values. The user can track the ways in which knowledge has been derived through inheritance.

- **Procedural attachment.** Programs may be associated with slots in the form of demons. Demons fire when slot values are accessed.

- **User-controlled search.** The user can control the way in which the system uses schemata as information.

- **Error handling.** A schema-based error handling facility permits the user to determine how the system will respond to errors.

- **Contexts.** Alternate problem solving viewpoints and knowledge base version management are provided by the context mechanism.

- **Integrated database system.** An sophisticated database system is provided to store schemata. Infrequently used schemata are automatically archived to memory. This feature is only available on the [...] systems.

Inference and Control Strategies

Knowledge-based systems have employed a variety of techniques to solve real-world problems. Since a single problem-solving technique has not yet proven adequate, Knowledge Craft provides the user with a powerful set of techniques which may be combined in a single application. Each technique is integrated with CRL.

- **CRL-OPS™.** Rule-based programming. The forward chaining capabilities of OPS-5 are integrated into the environment. Many successful expert systems used by industry today have been implemented in OPS-5

- **CRL-PROLOG™.** The system combines PROLOG inferencing with the representational power of CRL. Inheritance provides a supplemental inference method not available in other logic programming environments

- **Integrated object programming.** Large systems are more easily maintained when engineered with an object-oriented approach. Knowledge Craft supports a message-sending paradigm for invoking procedures. Objects are represented as schemata, methods as slots. Methods may be inherited or accessed from other objects. Procedures executed in reply to messages may be logic programs, rules or LISP functions

- **Multiple agenda manager.** A multiple agenda manager enables the user to schedule events against a real or simulated clock. Symbolic event-based simulations of complex real-world processes may be implemented using this mechanism

Interface Tools

Sophisticated graphic, menu and natural language interfaces may be constructed using Knowledge Craft's interface facilities.

- **Window Manager.** A device-independent window manager controls multi-window displays with a flexible viewport/window/canvas architecture.

- **2D graphics.** A device-independent graphics package enables the user to construct 2D graphic displays with scaling, rotation and zoom features. Schema-based icons can be used to convey input and output

- **Task manager.** Multiple tasks can be created, paused, resumed and aborted. Menu interfaces are easy to build using a hierarchical command system that includes pop-up menus, command completion and help facilities

- **Intelligent schema filler.** Simple knowledge acquisition interfaces can be built with this module. It allows customized schema-specific prompting for information

- **Natural language tools.** Language Craft™ may be used to construct natural language interfaces to Knowledge Craft applications.

Programming Workbench

Knowledge Craft provides a programming workbench for bit-mapped, mouse-controlled workstations. An icon-oriented interface enables system developers to move easily between specialized editors and monitoring windows. The Workbench includes fully-integrated Work Centers for CRL, CRL-OPS and CRL-PROLOG

- **CRL Work Center.** Knowledge base editors allow networks of schemata to be graphically displayed in tree form, new schemata to be added to the graph, and old ones deleted. Schemata can be opened to edit slots, values and meta-knowledge

- **CRL-OPS Work Center.** This full-screen interactive environment for developing and debugging CRL-OPS programs graphically captures program output and debugging information. It allows the user to alternate between running the program, examining its internal state, and editing OPS rules. A screen diagram displays changes to the state of the program, as well as textual debugging and tracing information

- **CRL-PROLOG Work Center.** This environment for developing and running CRL-PROLOG programs, provides facilities similar to the other Work Centers in the Programming Workbench. It traces the execution of CRL-PROLOG programs, and enables the developer to view program contents, enter queries, and modify program rules

Operating Environment

Knowledge Craft is implemented in Common LISP, and runs on Symbolics™ 3600 Series, TI Explorer™, VAX™ and MicroVAX™ hardware

Training and Technical Support

Each purchaser of Knowledge Craft is entitled to training and support services. An intensive two-week hands-on tutorial enables users to begin developing working knowledge-based programs

Introduction to Knowledge Craft™

Introduction to Knowledge Craft™

Knowledge Craft is a high productivity tool kit for knowledge engineers and AI system builders. Combining a feature-rich knowledge representation language with proven problem-solving techniques, Knowledge Craft dramatically reduces the effort required to build knowledge-based systems. In addition, sophisticated window, text, and graphics management modules are fully integrated with a command system interpreter, providing an interface buiding tool of significant power.

The goal of Knowledge Craft is to enable the cost-effective development of large-scale knowledge-based systems. Although designed for experienced system analysts and programmers, Knowledge Craft facilitates the development of front-end user interfaces for knowledge acquisition.

Highlights

- Frame-based knowledge representation language with procedural attachment and inheritance

- Database management

- Knowledge base editors

- Logic programming

- Rule-based programming

- Object-oriented programming

- Event management for simulation and scheduling

- Programmer's workbench

- Interface tools with 2D graphics and icons

- Complete Common Lisp integration

Applications

Knowledge Craft is now being used in many application areas:

- Project management

- Factory planning and scheduling

- Product selection

- Alloy design

- Computer-aided design

- Process diagnosis

- Long-range planning

- Distribution analysis

- Battlefield management

History and Philosophy

Knowledge Craft's approach is based upon proven experimental prototypes developed at Carnegie-Mellon University and used to solve diverse problems in over 25 industrial environments. Many of these systems are in commercial use today. Several conclusions resulting from this work influence the design of Knowledge Craft.

- Knowledge-based systems cannot be built using any one problem solving approach. Tool kits allowing flexible knowledge representations, and alternative problem-solving and control strategies are required.

- Tools must support rapid prototyping.

- Sophisticated interfaces are required to facilitate the man/machine transfer of information.

- Programming workbenches specialized for knowledge engineers are needed to enhance productivity.

- Database support for large knowledge bases must be provided.

- Systems must be built in a portable language to survive in a changing hardware environment.

CRL™: Carnegie Representation Language

- CRL™ is efficient, easy to use, and suitable for both large and small applications. The basic representational unit is a schema. Each schema has slots and values to store attributive and relational information about an entity. Relations link schemata to one another. Information in

one schema can be transferred to another using simple taxonomic inheritance or more sophisticated methods.

CRL provides functions for creating, deleting, and modifying schemata. The knowledge base editors discussed below allow user-friendly graphic access to these functions.

Figure 1 provides an example of a schema. A schema is composed of a schema name (printed in the bold font), a set of slots (printed in small caps) and the slot's values. Values can be any LISP expression. Symbols are used to reference schemata. When printed, a schema is enclosed by double braces with the schema name appearing at the top.[1]

```
(( make-cpu1-board-spec
        IS-A: engineering-activity specification-development
        SUB-ACTIVITY-OF: develop-board-cpu1
        INITIAL-ACTIVITY-OF: develop-board-cpu1
        EXPECTED-COMPLETION-DATE: "August 8, 1985"
        INITIATED: t
        COMPLETED: nil
        DESCRIPTION: "Develop specifications for the cpu board"
```

Figure 1: make-cpu1-board-spec Schema

The example in figure 1 comes from a knowledge base containing activity management information in respect to developing circuit boards. Taxonomic inheritance proceeds over IS-A relations. Thus in this case, all characteristics specification-development and engineering-activity are also true of make-cpu1-board-spec. Inheritance provides a means of "reasoning by default". Any values that are not specified in a schema may be inherited from another schema.

• User-defined relations and inheritance semantics. In order to facilitate world modeling, users may define new relations. Real world relations such as "has-project-leader", "sub-activity-of", and "has-sub-activity" may be created. For each relation, the user can specify inheritance semantics (i.e., information passing characteristics) indicating which slots and values can be inherited over that relation.

The relation SUB-ACTIVITY-OF, for instance, is user-defined. In this case, it is desirable that

---

[1] In the following text, schemata are indicated in bold; slots and relations in small caps; and values in italics.

any activity which is a SUB-ACTIVITY of another "inherit" its PROJECT-LEADER specification but not its DURATION. Figure 2 shows the develop-board-cpu1 activity. The make-cpu1-board-spec thus can be said to have *Jim_Smith* as its PROJECT-LEADER, since this slot is not specified in its schema. However, the duration of the whole project cannot be assumed to be the duration of any part; thus, the value in the DURATION slot will not be inherited over the SUB-ACTIVITY-OF relation.

```
{{ develop-board-cpu1

        IS-A: engineering-activity board-development

        SUB-ACTIVITY-OF: develop-computer

        HAS-SUB-ACTIVITY: make-cpu1-board-spec

        PROJECT-LEADER: Jim_Smith

        DURATION: (6 mo)

        DESCRIPTION: "Develop cpu board cpu1" }}
```

Figure 2: develop-board-cpu1 Schema

The above is one example of how information flow within the knowledge base can be controlled. CRL provides a rich language for describing the semantics of inheritance. Within this language, values may be included or excluded from inheritance under precise conditions. In addition, mapping operations may be applied during inheritance, converting one value to another as desired.

In additon to inheritance or information-passing properties, relations are defined by their "scope" and "transitivity." Scope is used to specify the nature of the objects between which the relation can hold. Transitivity is used to describe the nature of the links over which the relation is defined. For example, the SUB-ACTIVITY-OF relation can only hold meaningfully between objects which are activities. The transitivity of SUB-ACTIVITY-OF is such that any activity which is a SUB-ACTIVITY-OF another, is also a SUB-ACTIVITY-OF all activities of which the latter is a sub-activity. Thus, it can be said that make-board-cpu1-spec is also a SUB-ACTIVITY-OF develop-computer.

CRL is the only knowledge representation language within which relations can be fully defined, that is, in terms of inheritance semantics, transitivity, and scope. This makes it possible to use inheritance to make inferences that would otherwise require the use of rules.

• **Meta-knowledge representation.** Knowledge about the representation itself can be

systematically partitioned from knowledge about the domain being modelled. This facility is used to chronicle knowledge base development, maintain dependency information, model uncertainty, and provide "local specialization" (i.e., the tailoring of relation or slot c racteristics on a schema by schema basis).

In terms of the example under discussion, within the **develop-board-cpu1** schema, meta-knowledge would be used to show who entered the information that *Jim_Smith* was to be project leader. Within the **make-cpu1-board-spec** schema, it would be used to show that the value of PROJECT-LEADER was inherited from the **develop-board-cpu1**. Within this same schema, local specialization would be used to indicate that the any acceptable DURATION value must be less than 3 months. The certainty with which one expected to complete the activity on time is attached as meta-knowledge on values of the EXPECTED- COMPLETION-DATE slot.

Meta-knowledge may be associated with schemata, their slots, and values in the slots. It is represented by another schema, called a meta-schema, that is attached to the schema, slot, or value. Representing meta-knowledge as schemata provides a uniform approach to representation. The user is provided with access functions for retrieving meta-schemata. Once retrieved, they are manipulated just as any other schema.

- **Procedural attachment.** Functions may be associated with slots in the form of demons. Demons fire when slot values are accessed, that is, when an attempt is made to add, delete, or modify a value in a particular slot. In the example, under discussion, a demon is attached to the duration slot, such that when it is modified, a new estimate can be made of each activities EXPECTED-COMPLETION-DATE. Demon functions can be used to initiate or return control to CRL-PROLOG or CRL-OPS programs.

- **User-controlled search.** The user can control the way in which the system seeks out inheritable information. Large knowledge-based systems typically link each schema to a great number of others. A project management system under development at Carnegie Group has, for instance, over 100 relations defined in it. Thus, when values are to be inherited, there are many relations over which a value could be found in another schema. For example, unless told where to search, a knowledge representation system might erroneously seek project-leader information over a HAS-SUB-ACTIVITY rather than SUB-ACTIVITY-OF relation.

In providing a way to specify where to look for inheritable information, CRL can optimize on memory by avoiding duplication of values, while maintaining performance through intelligent search control. Nevertheless, inherited values may still be locally cached, if a user so desires.

User-controlled search can be used similarly to dynamically alter the semantics of the representation, or, in other words, to allow slots to play different roles. For instance,

make-cpu1-board-spec could be a SUB-ACTIVITY-OF a particular contract, say, the **abc_inc_contract**. When looked at in this way, in a matrix organization, the project leader might not be Jim Smith, who is the technical project leader, but rather Reed Jones, who is the contract coordinator. By specifying an inheritance path, the user can indicate if an access to PROJECT-LEADER should be interpreted from the technical or contract management point of view. The appropriate value would then be returned.

- **Error handling.** An integrated schema-based error handling facility permits the user to define how the system should react to errors. For instance, in our example, any attempt to set a duration of an activity that is greater than the activity it is part of will produce an error. The appropriate action to take can be specified in a schema -- with control passed back to the appropriate point in the program.

- **Contexts.** Alternate worlds reasoning and knowledge base version management are provided by the context mechanism. *Contexts* in CRL act as virtual copies of knowledge bases. In the copy, schemata can be created, modified, and destroyed without altering the original context. Contexts are structured as trees where each context may inherit the schemata present in its parent context. Hence, only schemata that are used in a context need be explicitly represented there. This avoids copying schemata that will never be used in the context. Schemata may be copied or moved across contexts.

  For instance, in order to explore the effects of completing the 'make-cpu1-board-spec late, a new context would be created in which the value of its duration slot would be altered. Any consequent changes to the ESTIMATED-COMPLETION-DATE of the activities of which it is part would occur in this new context. After analyzing the results, the context could be deleted without affecting the original knowledge base. If the change constituted an acceptable plan, the results could be merged back into the parent context.

- **Integrated database system.** A multi-user database system is provided to store schemata. Frequently-used schemata are automatically cached in memory; and automatically swapped out when the limits of the cache are reached. CRL keeps track of which schemata are in memory and which are in the database. The database provides a fast device-dependent mechanism for external storage. Alternatively, schemata can be saved in a device-independent source code form. These files can be dynamically created during a run.

Inference & Control Strategies

Knowledge-based systems have employed a variety of techniques to solve real-world problems. A single problem-solving technique has not yet proven adequate. Knowledge Craft provides the user with a powerful set of problem-solving techniques which may be combined in a single application. Each technique is integrated with CRL.

- CRL-OPS™. CRL-OPS is a superset of OPS-5, a forward chaining rule-based system. Many of the most successful expert systems used by industry today are implemented in OPS-5. CRL-OPS rules are if-then statements. For instance, a CRL-OPS rule is used to indicate that when an engineering activity is completed, each activity which is subsequent, or enabled by it, should be initiated.

```
(p enabling-rule
    ( <activity>
    (engineering-activity ^completed t ^schema-name <completed-activity> ))
    ( <next-activity>
    (engineering-activity ^enabled-by <completed-activity> ^initiated nil))

-->
    (modify <next-activity> ^initiated t ) )
```

Patterns on the left hand side of each CRL-OPS rule match all schemata which are "is-a" related to the class of the condition element. That is, the first pattern will match any schema in the knowledge base that IS-A engineering-activity and has a COMPLETED slot with the value t. The action of the rule will apply similarly to any schemata which IS-A engineering-activity, has the name of the first schema as the value of its ENABLED-BY slot and has not yet been initiated.

CRL-OPS rules can thus be written at the level of generality which is appropriate for the action of the rule. In this manner, rules may be written to apply either to a unique schema or a class of schemata. Rules will apply to schemata dynamically created at run time.

CRL-OPS is data-driven, so that when there is a change in any schema referenced by a rule, the rule-matcher is automatically notified. In addition, CRL-OPS supports left hand side functions, access to the CRL context mechanism, and integration with CRL-PROLOG. CRL-OPS rules are compiled into an efficient run time form.

- CRL-PROLOG™ combines PROLOG-like inferencing with the representational power of CRL. Prolog has been used heavily in Europe for over a decade. It has proved valuable as tool for building database query systems, as it provides a uniform mechanism for finding all or some schemata with particular characteristics that may be found in or inferred from a knowledge base.

For example, a query to find activities which will be completed after the initiation-date of an activity they enable would be:

Query: (late-activity ?activity ?date)

Assuming the following axiom was part of a loaded logic program:

(late-activity <x> <t2>) < (<x> ^is-a activity)

```
( <x> ^expected-completion-date <t1> )
( <y> ^enabled-by <x> ^initiation-date <t2> )
( < <t2> <t1>)
( bind <t2> (print-time '<t2> )
```

This axiom says that an activity <x> is late in respect to another activity <y>, if <y> is enabled by <x>, and the completion date of <x> is greater than the initiation date of <y>. The axiom returns the binding of <x> and the INITIATION-DATE of <y> which indicates the date that <x> ought to be completed by. The bind predicate is provided to enable call-outs to Common LISP. An ops-like syntax is used to refer to schemata; a prefix predicate syntax is used to define arbitrary facts.

Query: (late-activity ?activity ?date)

==>

?activity = develop-board-cpu1
?date   = August 5, 1985

CRL-PROLOG is a backward chaining language. This means that if a goal (or query) cannot be inferred in terms of the schemata already available, an attempt will be made to infer it from axioms that apply to each clause in the body of the axiom, whose head matches the query. For example, if no schema matched the clause (<y> ^enabled-by <y>), CRL-PROLOG would attempt to infer this result from axioms that could prove that one activity was enabled-by another. CRL-PROLOG, thus, provides Knowledge Craft with a general sub-goaling mechanism. Using CRL-PROLOG and CRL-OPS together blackboard control architectures can be designed that integrate goal-driven and data-driven problem-solving strategies.

CRL-PROLOG uses an OPS-like infix syntax to refer to schemata. Arbitrary facts may be defined in addition, using a predicate postfix notation.

- Integrated object-oriented programming. Large systems are more easily maintained when engineered with an object-oriented approach. Knowledge Craft supports a message-sending paradigm for invoking procedures. Objects are represented as schemata; methods as slots. Methods may be inherited or accessed from other objects. Procedures executed in reply to messages may evoke CRL-PROLOG or CRL-OPS, monitor displays, or call arbitrary LISP functions.

For example, each activity schema in the knowledge base under discussion has a list of people working on that activity. The schema for each person includes a list of activities they are working on, and the due dates for each activity. Within the schemata activity and

person, there is a slot called PRINT-SCHEDULE whose function is to print a schedule to the monitor screen.

```
(( activity
    PRINT-SCHEDULE: activity-print-schedule ))

(( person
    PRINT-SCHEDULE: person-print-schedule ))
```

For activities, the LISP function 'activity-print-schedule prints a table providing the name of the person working on the activity, what they are doing, and their estimated completion date. For persons, the 'person-print-schedule function prints out all activities a person is working on, their particular task, and an estimated completion date. These functions are executed by a simple function:

(send-message 'Jim_Smith 'print-schedule)

would execute the function in the 'print-schedule slot of the 'jim_smith schema. If there were no value in this slot, it would be inherited from the person schema over the 'is-a relation.

(send-message 'develop-board-cpu1 'print-schedule)

would execute the function in the PRINT-SCHEDULE slot of the develop-board-cpu1 schema. If there were no value in this slot, it would be inherited from the activity schema over the IS-A relation.

The send-message function allows reference to a standard method, rather than the function name that implements that method for a particular schema or class of schemata. Programming modularity is thus achieved both by hiding low level details, and by the inheritance mechanism which allows methods to be described at the right level of abstraction.

* Event management. Possible world reasoning, or simulation, requires both a mechanism for distinguishing different worlds, and a mechanism for scheduling the realization of events. Knowledge Craft provides both. While the context mechanism of CRL provides contexts, the multiple agenda manager enables the user to schedule events against a real or simulated clock. Symbolic event-based simulations of complex real-world processes are implemented using this mechanism.

System Building Tools

Interface Tools

Knowledge-based systems typically require sophisticated user interfaces. New knowledge generated by the program must be informatively displayed. Powerful devices must be provided for requesting and accepting inputs of considerable variety. Knowledge Craft supplies the knowledge engineer with tools that reduce the programming effort required to build application interfaces.

- **Window manager.** A device-independent window manager controls multi-window displays with a flexible viewport/window/canvas architecture. A canvas is an infinite display space with a user-definable coordinate sytem. A window is a section of the canvas. A viewport is a projection of a window onto a monitor display. A tripartite architecture of this type optimizes for both flexibility and performance. Scrolling and 2-dimensional transformation is provided at each level.

- **2D graphics.** A device-independent graphics package enables the user to construct 2D graphic displays with scaling, rotation, and information zoom features. Information zooming allows composite symbols to be unpacked, displaying more information in response to mouse pointing. Scaling provides both single item and window transformations.

    The basic graphic primitives -- string, line, circle, rectangle, box, polygon, and spline -- are included. CORE standards are followed closely. Schema-based icons can be used to convey input and output. Editors facilitate the creation and inspection of graphics items. Both mouse pointing and display grids facilitate the exact positioning of items.

- **Task manager.** Multiple tasks can be created, paused, resumed, and aborted. Windows can be associated with tasks allowing the creation of workbenches with the capability to switch back and forth between particular tasks. Each task can control a tree of menus.

    Menu interfaces are easy to build using a hierarchical command system that includes pop-up menus, mouse pointing, multi-word spelling completion, and help facilities. An emacs-like editor is provided for editing input.

- **Intelligent schema filler.** Simple knowledge acquisition interfaces can be built with this module. It allows customized schema-specific prompting of information.

Programming Workbenches

In order to increase the productivity of knowledge engineers and system developers, Knowledge Craft provides a programming workbench. While the workbench is oriented to bit-mapped, mouse-controlled workstations, many capabilities are also available for alpha-numeric terminals.

- **Knowledge Craft Shell.** The Shell enables developers to move easily between knowledge base editors, debugging environments, and a running program. An icon-oriented interface represents tasks which can be activated by the mouse. New icons can be added to represent user applications. Escape to the local machine environment is typically supported.

- **Knowledge base editors.** Networks of schemata may be graphically displayed in tree form. New schemata can be added to the graph; old ones deleted. Networks may be scrolled. schemata can be opened for editing slots and values. An emacs-like editor is provided.

  Since the editor knows about user-defined relations, it provides a browsing capability, interpreting any legal path description through the network. The editor can be multiply instantiated allowing simultaneous editing of different parts of the knowledge base, including the context tree.

- **Debugging environments.** Specialized window-oriented debugging facilities are provided for CRL-OPS and CRL-PROLOG. For CRL-OPS, trace windows are provided for watching changes to working memory and the conflict set. Display items are active. For instance, each rule in the conflict set is an active item. Mouse pointing is used to determine matches for that rule, edit the rule body, and remove it from the conflict set, among other things. For CRL-PROLOG, a trace window of the goal tree is provided. Each node of the tree can be expanded to show the clause which generated the goal, and the instantiations of relevant variables.

Operating Environment

Knowledge Craft is implemented in Common LISP and is portable to most machines with a Common LISP implementation. It is compatible with other Carnigie Group products, such as Language Craft[TM]. which may be used to build natural language interfaces.

Training

An intensive two-week hands-on tutorial enables users to become proficient in developing knowledge-based programs using the package. Knowledge Craft is best used by experienced programmers. The expectation is that knowledge-based systems will be built by a mixed team of software specialists, knowledge engineers, and domain experts.

A Common LISP course is also available.

[TM] Knowledge Craft, CRL, CRL-OPS, CRL-PROLOG and Language Craft are trademarks of Carnegie Group Inc.

APPENDIX B

PRICE INFORMATION FOR THE

"ART" EXPERT SYSTEM DEVELOPMENT TOOL

Inference

October 23, 1986

Mr. Charles Ziegler
Delta Information Systems
Horsham Business Center
Building 3
300 Walsh Road
Horsham, PA 19044

Dear Chuck:

I am delighted to hear Delta is preparing to acquire a copy of
the Automated Reasoning Tool, (A.R.T.). Inference Corporation
is anxious to provide Delta with the best support available in
the expert systems marketplace.

In that regard, let this letter document my recommendation for
the preferred development environment. You have specified
A.R.T./C. Code to be delivered to Delta by January 1987. That
requirement is acceptable to Inference. Therefore, Inference
recommends your order specify A.R.T./C. Code rather than
A.R.T./Lisp Code, with the target hardware being a Dec A.I.
Workstation.

Also, be aware that your runtime modules can be deployed in
A.R.T./C. Code and will be over 99% garbage free. The following
discount schedule will apply to the runtime modules shipped by
Delta Information Systems:

Copies	Pricing
1 - 10	$8,000
11 - 50	7,000
51 - 100	5,000
101 - 200	3,000
201 - 300	2,000
301 - Up	1,500

In closing, please feel free to contact me at the Marlton Office
should you have further concerns or questions.

Regards,

Dennis Hartigan
Inference Corporation
Marlton, New Jersey

DH/klb

B - 1

Inference

To:
Chuck Ziegler
Delta Information Systems
Horsham Business Center
Building 3
300 Welsh Road
Horsham, PA 19044

Inference Corporation

5300 W Century Blvd
Los Angeles
California 90045

213 417-7997

NO.	132.-DH
Date	15 October 1986
P.O. Deadline	31 December 1986

The price shown in this Quotation shall not be increased for thirty (30) days from the date shown above.

License Agreement must be executed and returned to Inference Corporation prior to receipt of software. Terms: Net 30 days from invoice date

Item	Description	Price	Amount
	ART - AUTOMATED REASONING TOOL		
1	ART: (copy 1) LISP VERSION 5 Days Knowledge Engineering 2 Sets of ART Documentation 90 Days Maintenance & Technical Support - Full access to Inference Hotline - New Version & Documentation Updates	$65,000.00	
2	TRAINING: Per person/per week Week One - Introduction to ART Week Two - Viewpoints	2,500.00	
3	KNOWLEDGE ENGINEERING: Per day, plus expenses (Charged in 1/2 day increments 4 hours = 1/2 day)	1,000.00	
4	MULTIPLE COPY DISCOUNT SCHEDULE:		

Copies	Price	KE Included
2 - 5	45,000	4 days
6 - 10	38,000	3 days
11 - up	29,500	2 days

Item	Description	Price	Amount
5	ANNUAL MAINTENANCE & TECHNICAL SUPPORT: (Commences 90 days after shipping, billed quarterly, payable in advance).	7,500.00	

By

Inference

N° 1330

To:

Chuck Ziegler
Delta Information Systems
Horsham Business Center
Building 3
300 Welsh Road
Horsham. PA 19044

Inference Corporation

5300 W Century Blvd
Los Angeles
California 90045

213 417-7997

NO.	1330-DH
Date	15 October 1986
P.O. Deadline	31 December 1986

The price shown in this Quotation shall not be increased for thirty (30) days from the date shown above.

License Agreement must be executed and returned to Inference Corporation prior to receipt of software. Terms: Net 30 days from invoice date

Item	Description	Price	Amount
	ART -- AUTOMATED REASONING TOOL		
1	* ART: (copy 1 "C" Version/Delivery before 12/31/86) 5 Days Knowledge Engineering 2 Sets of ART Documentation 90 Days Maintenance & Technical Support - Full access to Inference Hotline - New Version & Documentation Updates	$45,000.00	
2	TRAINING: Per person/per week Week One - Introduction to ART Week Two - Viewpoints	2,500.00	
3	KNOWLEDGE ENGINEERING: Per day, plus expenses (Charged in 1/2 day increments 4 hours = 1/2 day)	1,000.00	
4	MULTIPLE COPY DISCOUNT SCHEDULE:		

Copies	Price	KE Included
2 5	45,000	4 days
6 - 10	38,000	3 days
11 - up	29,500	2 days

Item	Description	Price	Amount
5	ANNUAL MAINTENANCE & TECHNICAL SUPP . (Commences 90 days after shipping, billed quarterly, payable in advance).	7,500.00	
	* P.O. and signed license agreements must be received prior to 31 December 1986 to qualify for discount.		

By *Donald L. Hammon*

ART

PRODUCT LICENSE AGREEMENT

Between

Inference Corporation
5300 W. Century Boulevard
Los Angeles, CA 90045
(213) 417-7997

And

_____

_____

_____

_____

DATE _____

NO. _____

EXHIBIT A

1   Identification of CPUs upon which the ART binary code may be used

    (a) Mfgrs. Name _____

    (b) Serial No _____

    (c) Room No. or Mail Sta _____

    (d) Address _____

                 _____

                  _____

2   Licensee Purchase Order Number _____

3   Invoice to _____

    Address _____

             _____

             _____

EXHIBIT B

Inference Corporation

Maintenance and Enhancement Plan
(Effective September 1 1986)

A. A Maintenance and Enhancement Plan (hereinafter referred to as the "Plan") is available to Licensee as specified in the following paragraphs. The Plan includes the following provisions

1. Customer Support

To supply a reasonable amount of customer telephone support via Inference's "Hotline" during the period of 6 a.m to 5 p.m. Pacific Standard Time/Pacific Daylight Savings Time. Monday through Friday. excluding Inference's observed holidays On-site support may be provided at a fee to be agreed upon by both parties

2. Modifications, Enhancements and Updates

To provide published feature modifications, enhancements and update releases which Inference. at its discretion. deems to be logical improvements to the original Products supplied to Licensee. The foregoing does not include providing new Products to Licensee.

B. The provisions of the Plan are contingent upon the following 1) A current and valid license for the Products. 2) The Products being unmodified (except as modified by releases developed and provided by Inference) and maintained at the latest release level: and 3) The hardware on which the Products are installed and used containing the configuration properly maintained and at the latest generally released revision level as specified by Inference for installation and use of the Products.

C. **Fees** If Licensee has a permanent license, or a license with monthly payments. Licensee is subscribed in the Plan, at no cost. for a period of ninety (90) days following the effective date of the Product License Agreement (the "Agreement") Prior to the end of the aforementioned ninety (90) day period. Licensee shall be invoiced the fee for the first charged year of the the Plan and shall pay such fee quarterly in advance, for each copy initially licensed. Thereafter on each anniversary date of the effective date of the Agreement. Licensee's subscription in the Plan shall be automatically renewed (and shall be subject to the same invoicing and payment terms as stated in the preceding sentence) unless Inference receives written notice of Licensee's intent to cancel its subscription in the Plan thirty (30) days prior to Licensee's anniversary date of the effective date of the Agreement Licensee may. at a later time. renew its subscription and receive the benefits of the Plan upon payment of the annual fee for the Plan in effect at the time of renewal plus a reinstatement fee equal to twenty percent (20%) of the annual fee for the Plan in effect at the time of renewal

Inference's fee for the Plan for the initial one (1) year charged period shall be the fee as stated in Inference's then current published price list. The fee(s) may be increased each year on the anniversary date of the effective date of the Agreement by the lower of the following: Six percent (6%) or the Consumer Price Index ("CPI") as applied against the fee that Licensee paid for the previously charged year The CPI that shall be used is the CPI that has been published by the Bureau of Labor Statistics for the Los Angeles/Long Beach Area on the date. or closest to the date. that Licensee's participation in the Plan must be renewed. INFERENCE RESERVES THE RIGHT TO MODIFY THE FEE INCREASES DESCRIBED ABOVE AT ANY TIME. BY ANY AMOUNT

D **Taxes and Other Charges** The payments set forth in Section C are exclusive of all tariffs. duties. sales taxes. use taxes and like levies or taxes, and all of the foregoing shall be borne by Licensee Any such levies. taxes or charges which Inference may be required to pay on behalf of Licensee shall be billed to Licensee when incurred by Inference and shall be due and payable when billed

E. **Method of Shipment** Items supplied under this Agreement will be shipped F O B Inference's Los Angeles. California facility In the absence of instructions to the contrary. Inference. on behalf of Licensee. will select the carrier. but shall not be deemed thereby to assume any liability in connection with the shipment. nor shall the carrier be construed to be an agent of Inference Costs of shipment insurance and handling will be the responsibility of Licensee

F **Title** Title to and risk of loss of items supplied under this Agreement shall pass to Licensee upon delivery to the carrier at the F O B point

G **Packaging** Items supplied under this Agreement will be shipped in Inference's standard packaging

This Agreement effective this _____ day of _____, 198__ by and between Inference Corporation, a corporation of the State of California, located at 5300 West Century Boulevard, Los Angeles, California, 90045, hereinafter referred to as "Inference"; and _____, a corporation of the State of _____, located at _____

_____ hereinafter referred to as "Licensee".

WHEREAS, INFERENCE has developed a computer program which is an Automated Reasoning Tool, which will hereinafter be referred to as "ART" and whereas the ART program is capable of developing useful inferences and analyses from available input information.

WHEREAS, INFERENCE has copyright protection on the ART program, and has trademark rights in "ART" for computer programs, and

WHEREAS, INFERENCE wishes to grant a Permanent License to use the ART program as more fully set forth below

NOW THEREFORE, in consideration of the foregoing premises, and the mutual covenants set forth below INFERENCE and LICENSEE hereby agree as follows

1. License

Inference grants and Licensee accepts on the terms and conditions contained in this Agreement a non-assignable, non-transferable, non-exclusive license to use the ART proprietary computer programs and related materials ("Products") on the CPUs and at the locations specified in Exhibit A, a copy of which is attached hereto and incorporated herein by reference.

2. Title

Inference warrants and represents that it is the owner of the Products. Title and full ownership rights to the Products shall remain the sole property of Inference. Licensee acknowledges, understands and agrees that the Products constitute valuable proprietary assets and trade secrets of Inference embodying substantial creative efforts and confidential information.

3. Scope of Use, Terms and Fees

A. The Products may be used by the Licensee only for Licensee's own internal use at the Licensee location and on the CPU as designated in Exhibit A (the CPU listed in Exhibit A is hereinafter referred to as "the Authorized CPU"). Licensee may not change the location of the use of the Products and may not use the Products or copies thereof on any CPU other than the Authorized CPU without the prior written consent of Inference, which will not be unreasonably withheld.

4. License Fees

Licensee agrees to pay to Inference, net thirty (30) days from the date of invoice the sum of _____

_____ $_____ for a Permanent License to use the ART binary code. Inference's standard software maintenance support and updates for ART are included in the above price for a period of ninety (90) days following the date of signing of this Agreement. If such fees are not paid in a timely manner, interest will be charged at the greater of 1.5% per month or the maximum allowed by law.

5. Type of License and Usage Period

The Products may be licensed on a permanent license basis for a one-time fee as specified above. The usage period of a permanent license is perpetual, subject to termination in accordance with the terms of this Agreement.

6. Confidentiality of Proprietary Information

A. Licensee expressly acknowledges, understands and agrees that the Products contain confidential information and other data proprietary to Inference. Licensee agrees not to allow any such confidential information or data to be disclosed or reproduced except for use in accordance with Paragraph 3 and as provided in sub-paragraph D below. Licensee further agrees not to allow any machine readable version of the Products to be printed, listed, decompiled, disassembled or reverse engineered.

B. Licensee, its agents, contractors and employees, agree to maintain all information and data contained in the Products including proprietary computer programs, documentation, generated output, modifications and conversions, in strict confidence for Inference. Licensee agrees to protect the Products using at least the same degree of care it uses to protect its own proprietary and confidential data of like importance, but in no event shall such care be less than a reasonably prudent business person would take in a like or similar situation. Licensee agrees to restrict access to and display of such information and data to such Licensee personnel who (i) have a need to have such access or see such display to enable Licensee to utilize the Products as contemplated by this Agreement and (ii) have been advised of and have agreed to treat the Products and such information and data in accordance with this Paragraph 6.

C. Licensee's obligations contained in this Paragraph 6 are of a special and unique character which give them a peculiar value to Inference and Inference cannot be adequately compensated in damages in an action at law in the event Licensee breaches such obligations.

Licensee therefore agrees that, in addition to any other remedies which Inference may possess, Inference shall be entitled to injunctive or other equitable relief in the form of a preliminary and permanent injunctions or other appropriate or similar equitable remedies, in the event of an actual or threatened breach of said obligations by Licensee.

D   Licensee agrees to reproduce and include Inference's proprietary and copyright notice on any copies, in whole or in part, in any form, of the Products, including but not limited to, reproduction of the international copyright notice consisting of a "c" within a circle followed by the appropriate year of copyright as designated by Inference and the name "Inference Corporation" in form similar to the following

> "COPYRIGHT© 1984 INFERENCE CORP AN UNPUBLISHED WORK —THE ART™ PROGRAM IS THE SUBJECT OF TRADE SECRETS AND COPYRIGHTS LICENSED FROM INFERENCE CORP USE OR DISCLOSURE OF THE ART PROGRAM TO UNAUTHORIZED PERSONS IS PROHIBITED

E   The provisions of this Paragraph 6 shall survive the termination of this Agreement

7. Information Similar to Proprietary Information

Licensee agrees to abide by the provisions of Paragraph 6 with respect to information and data provided by Inference irrespective of whether Licensee rightfully possesses identical or similar information or data obtained rightfully from sources other than Inference who had the right to disclose such information and or data Licensee also agrees that if Licensee becomes aware that it possesses information or data that is identical or similar to that which Inference treats as proprietary, Licensee will give Inference prompt written notice thereof and the source from which it was obtained

8. Proprietary Rights Indemnification

Inference warrants that the Products do not infringe upon or violate any United States patent, copyright or trade secret Inference will defend at its expense any action brought against Licensee to the extent that it is based on a claim that Products used within the scope of the license hereunder infringe a United States patent, copyright or trade secret and will pay any costs and damages finally awarded against the Licensee in such action which are attributable to such claim, subject to the limitation of liability stated in this paragraph 8, provided that Licensee notifies Inference promptly in writing of the claim, allows Inference to fully control the defense of such claim and does not agree to any settlement of such claim without Inference's written consent Should the Products become, or in Inference's opinion be likely to become, the subject of any claim of infringement, Inference may procure for the Licensee the right to continue using the Products, replace or modify them to make them non-infringing or discontinue the license of them Inference shall have no liability for any claim of infringement based upon (i) use of other than the latest unmodified release of the Products made available to Licensee by Inference if such infringement would have been avoided by the use of such release of the Products, (ii) use or combination of the Products with non-Inference programs or data if such infringement would not have occurred without such use or combination or (iii) use of the Products after receiving notice, that the Products infringe a trade secret of a third party unless prompt written notice thereof is given Inference The foregoing states the entire liability of Inference with respect to infringement of any patents, copyrights or trade secrets by the Products and Inference shall have no liability with respect to any other proprietary rights

9. Liability

Except as specified in this Agreement, Inference shall not be liable for any loss or damage that may arise in connection with the furnishing to or use by Licensee of Products, or the performance of the Products and in no event shall Inference be liable for any indirect, special, incidental or consequential damages Except as otherwise specified in Paragraph 8 and Paragraph 10, Licensee shall not be entitled to any monetary damages against Inference in excess of the amounts paid to Inference by Licensee hereunder No action, regardless of form, arising out of the transactions under this Agreement, may be brought by either party more than two (2) years after the cause of action has accrued, except that an action for non-payment may be brought within two years after the date of last payment

10. Warranty

Inference warrants that at the time of delivery of the original Products supplied to Licensee and for a period of ninety (90) days thereafter, the original Products will be in substantial accordance with specifications in the applicable technical reference manual The extent of Inference's liability under this warranty shall be limited to the correction or replacement as soon as practicable of any substantial deviation in the original Products (or any subsequent releases of Products) from the specifications in the applicable technical reference manual, which Inference reasonably determines to be necessary, at Inference's own cost and expense, provided written notice of such substantial deviations is received by Inference during the warranty period This warranty shall not apply if (i) the Products (or parts thereof) shall not be used in accordance with Inference's instructions, (ii) the Products shall have been altered, modified or converted by Licensee without the written approval of Inference, (iii) any of Licensee's equipment shall malfunction which results in the Products not performing in accordance with the specifications in the applicable technical reference manual, or (iv) other cause within the control of Licensee shall result in any part of the Products becoming inoperative or substantially deviating from the specifications in the applicable technical reference manual THE FOREGOING WARRANTY IS IN LIEU OF OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

11. Maintenance Plan

The terms and conditions of Inference's Maintenance and Enhancement Plan are to be found in Exhibit B a copy of which is attached hereto and incorporated herein by reference

12. Delays

Inference shall not be liable for delays in the performance of its obligations hereunder due to causes beyond its reasonable control including, but not limited to, Acts of God, strikes or inability to obtain labor or materials

13. Notification

All notices which any party may be required or desire to give to any other party shall be given by personal service, registered mail or certified mail to the other party at his respective address set forth on the front of this Agreement. Mailed notices shall be deemed to be received on the fifth California business day following the date of mailing.

14. Successors

This Agreement, together with all schedules or modifications now and hereafter made a part hereof shall be binding on the respective parties and their respective heirs, executors, administrators, legal representatives, successors and assigns.

15. Governing Law

This Agreement shall be governed by the laws of the State of California applicable to contracts wholly executed and wholly to be performed within the State of California. This Agreement constitutes the entire agreement and understanding between the parties relating to the license and use of the Products, and all other prior agreements, arrangements or understandings, oral or written, are merged into and superseded by the terms of this Agreement. Title and Paragraph headings are for convenient references and are not a part of this Agreement

16. Invalid Provisions

No waiver of any breach of this Agreement shall constitute a waiver of any other breach of the same or other provisions of this Agreement and no waiver shall be effective unless made in writing. In the event that any provisions herein shall be illegal or unenforceable, such provisions shall be severed and the entire Agreement shall not fail, but the balance of the Agreement shall continue in full force and effect.

17. Term

This Agreement shall commence on the date of execution hereof and it shall remain in force until the licenses of all Products have completed their specified usage periods.

18. Termination

Upon completion of a specified usage period for a license of a Product, or if Licensee fails to fulfill its obligations under this Agreement, Inference may upon its election and in addition to any other remedies it may have, upon written notice to Licensee of the breach, and failure by Licensee to cure such breach within two (2) weeks, terminate all of the rights granted by it hereunder. Upon termination as set forth herein, Licensee shall, within two (2) weeks, return to Inference the Products supplied to Licensee and destroy or delete all copies of the Products, including but not limited to any inference supplied information, load modules, back-up or archival information data sets and documentation. Licensee will verify this action, in writing to Inference.

IN WITNESS WHEREOF, each of the parties has caused this Agreement to be executed in duplicate originals by its duly authorized representatives on the respective dates entered below.

Agreed and Accepted   Agreed and Accepted

_____ Inference Corporation _____   LICENSEE _____

SIGNATURE _____   SIGNATURE _____

NAME _____   NAME _____

TITLE _____   TITLE _____

DATE _____   DATE _____

APPENDIX   C

PRICE INFORMATION FOR THE S.1

EXPERT SYSTEM SOFTWARE

Teknowledge Inc
1717 Abbey Oak Drive
Vienna, Virginia 22180
(703) 255 3385

TEKNOWLEDGE

Applied Artificial Intelligence

Ronald K. Goldstein
Senior Account Manager

October 21, 1986

Mr. Chuck Zeigler
Delta Information Systems
Horsham Business Center
Building 3
300 Welsh Road
Horsham, Pa. 19044

Dear Mr. Zeigler:

Teknowledge, Inc. is pleased to provide you with a quote for
MicroVAX based S.1 expert system software. S.1 is available for both
VMS and Unix (Ultrix) operating systems and can be delivered on either
cartridge or disk media.

Teknowledge offers the initial copy of S.1 and the following
additional items for a total of $25,000 for the MicroVAX version of
S.1:

- o S.1 expert system development software for your work-
  station;
- o S.1 Documentation (Users Guide, Reference Manual and
  Sample Knowledge Systems);
- o The S.1 packager used to build and field delivery
  systems;
- o One week of Knowledge Engineering Methodology(KEM)
  training;
- o Two weeks of S.1 training; and
- o One year of telephone hotline support and S.1 product
  updates.

Additional development copies of **S.1** are available at $25,000
through the 5th copy and decrease to $20,000 thereafter.

Delivery (or run-time) licenses of S.1 applications are priced at
$3,000 for the first through the 30th copy and $2,700 per copy for
the 31st through the 100th copy. Teknowledge is open to working with
Delta Information Systems in working out special pricing for
situations involving a commitment to purchase blocks of development
and/or delivery copies of S.1.

If you have any further questions, please do not hesitate to call.

Sincerely,

Ron Goldstein

TEKNOWLEDGE, INC.
Software License Agreement
Paragraph A

This License Agreement between Teknowledge, Inc. and the Licensee identified below consists of this Paragraph A and the Teknowledge, Inc. General Terms and Conditions Reference Number: _____51011_____. which is incorporated by this reference.

Date_____

Purchase Order Number_____ ____

The Product(s) to be provided are:

Quantity	Product	Designated CPU Make & Serial #*	Maintenance Period	Maintenance Fee	License Fee
_____	S.1 (DEC VAX)				
_____	S.1 (Xerox)				
_____	S.1 (Symbolics)				
_____	M.1				
_____	M.1 Delivery Systems				
_____	M.1a				

*Serial number required for S.1 licenses only.

Total Fees: $_____ _____ (exclusive of applicable taxes and shipping charges).

Site(s) location (not required for M.1 Delivery Systems):

(a)_____
   Street Address   City   State   Zip Code

(b)_____
   Street Address   City   State   Zip Code

Accepted by LICENSEE:

Company Name:_____

Company Address:_____

_____

Signature:_____

Name_____

Title_____

Date_____

Accepted by TEKNOWLEDGE, INC.:

TEKNOWLEDGE INC.
525 University Avenue, Suite 200
Palo Alto, CA 94301

Signature_____

Name_____

Title_____

Date_____

TEKNOWLEDGE, INC.
S.1 General Terms and Conditions

51041

Reference Number:

1. DEFINITIONS

a. Agreement. This Agreement consists of these General Terms and Conditions and Paragraph A attached hereto. These General Terms and Conditions are general terms and conditions for the licensing of Teknowledge's proprietary computer software products. More than one Paragraph A may incorporate these General Terms and Conditions by reference. Each Paragraph A, taken together with these General Terms and Conditions, shall constitute a separate Agreement and shall be considered independent of any other agreements between the parties which incorporate these General Terms and Conditions.

b. Product. The term "Product" means one or more of the proprietary computer software programs identified in Paragraph A, all related materials, documentation and information received by Licensee from Licensor and the published specifications for the Product. Paragraph A may identify more than one Product or more than one copy of any Product.

c. Designated CPU. The term "Designated CPU" means any central processing unit, including its associated peripheral units, described in Paragraph A. Paragraph A may designate more than one CPU.

2. LICENSE

a. Grant of License. Licensor hereby grants to Licensee and Licensee hereby accepts from Licensor a nonexclusive, nontransferable license to use the Product(s) in accordance with this Agreement during the term specified in Section 3. Licensee is aware that this Agreement grants Licensee no title or rights of ownership in the Product and that Licensor considers the Product to be the proprietary information and a trade secret of licensor. It is expressly understood and agreed that the obligations of this section shall survive the expiration or termination of the Agreement or any provision hereof.

b. Payment of License Fee. As a condition to Licensor's obligations hereunder, Licensee agrees to pay, within thirty (30) days following receipt of an invoice by Licensor, the one-time License Fee set forth for such Product in Paragraph A.

c. Restrictions on Use. Licensee is authorized to use the Product only for Licensee's internal purposes and only on the Designated CPUs at the site(s) specified in Paragraph A. Licensee agrees that it will not use or permit the Product to be used in any manner, whether directly or indirectly, that would enable Licensee's customers or any other person or entity to use the Product on other than a Designated CPU. Licensee acknowledges and agrees that this Subsection 2(C) specifically prohibits, without limitation: (i) incorporation of the Product into any of Licensee's products intended for distribution, or actually distributed, to third parties, (ii) use of the product on or over any computer network system, except one such use for the purpose of initial installation of the Product on the Designated CPU, or thereafter, reinstallations following system failures, (iii)disassembly or copying (except preparation of a single back-up copy) of the Product or (iv) sublicense, commercial time-sharing or rental of the Product. Licensee may not assign or sublicense the license granted hereunder without the prior written consent of Licensor, and any such purported assignment or sublicense shall be void.

d. Use on Other than Designated CPU. Notwithstanding the foregoing restrictions on use, but subject to the requirements of notice and consent set forth in this Subsection 2(D), Licensee may use the Product on other than a Designated CPU in the following circumstances: (i) if a Designated CPU cannot be used because of equipment or software malfunctions, Licensee may temporarily use the Product on another CPU; and (ii) if a Designated CPU is replaced by Licensee, Licensee may designate a successor Designated CPU and use the Product on such CPU, provided that use on the prior Designated CPU has ceased. Licensee must give Licensor prior written notice and Licensor must give its prior written consent (which shall not be unreasonably withheld) before such other uses are permitted.

e. Documentation. Licensor shall provide Licensee with: (i) the Product in machine readable form and, (ii) Licensor's then-current user documentation for the Product ("Documentation").

f. Proprietary Markings. License agrees to maintain all copyright, and/or other markings and/or legends

placed upon, contained with or included in the Product, documentation and/or related materials. Such obligation to maintain shall also include the obligation to perpetuate such markings and/or legends on all copies of the Product, documentation and/or related materials prepared by Licensee in accordance with the terms of this Agreement.

3. TERM AND TERMINATION

a. The license granted under this Agreement shall commence upon the delivery of the Product to Licensee and shall continue in perpetuity unless sooner terminated pursuant to Subsection 3(B) below.

b. In the event of any default by the Licensee of any material term covenant, or obligation under this Agreement, this Agreement shall terminate on thirty (30) days prior written notice by Licensor.

c. The following conditions apply upon termination: (i) the Licensee shall discontinue use of the Product and shall deliver to Licensor the Product and related materials furnished by Licensor together with all copies thereof, and shall warrant in writing, within thirty (30) days of termination, that the Product, related materials and all copies thereof have been returned to Licensor; and (ii) the Licensee shall also erase or destroy any of the Product contained in computer memory or data storage apparatus under the control of the Licensee and shall remove the Product from all software systems of Licensee. The Licensee shall warrant such in writing to Licensor within thirty (30) days of termination.

d. If Licensee fails to comply with the provisions of Subsection 3(C) above, Licensor shall have the right to take possession of the Product, wherever the same may be located, upon notice and demand, in accordance with process of law.

4. MAINTENANCE AND SUPPORT

a. Maintenance and Support Services. Subject to the terms, conditions and charges set forth in this section, Licensor will provide Licensee with maintenance and support services for the Product(s) for a period of ninety (90) days from date of invoice as is necessary to cause the Product(s) to perform in accordance with its current published specifications. Licensee acknowledges that access by Licensor to the Designated CPU and Licensee's products being developed with, or containing, the Product(s) shall be essential to performance by Licensor of its obligations under this Section. The maintenance and support services shall be performed by telephone from Licensor's offices in Palo Alto, California or at Licensee's location, as Licensor may elect in its sole discretion.

b. Duration of, and Charges for, Maintenance and Support. There will be no additional charge for maintenance and support during the first ninety (90) days. For each year after the first ninety (90) days Licensor will continue to provide Licensee with maintenance and support services as described in Subsection 4(A) above, provided that Licensee executes a separate Maintenance Agreement and pays Licensor in advance the annual maintenance and support charges then in effect.

c. Limitations on Licensor's Obligations.

Licensee understands and agrees that Licensor may develop and market new or different computer programs which use part or all of the Product(s) and which perform all or part of the functions performed by the Product(s). Nothing contained in this Agreement gives Licensee any rights with respect to such new or different computer programs. Any failure by Licensor to provide ongoing annual maintenance and support on the anniversary date of the agreement shall not constitute grounds for terminating this Agreement but shall be only a basis for terminating the parties' future obligations with respect to maintenance and support.

d. Licensee Confidential Information.

Licensor understands that it may, in the course of performing the maintenance and support services, have access to certain confidential information of Licensee which Licensee has clearly marked as such or otherwise identified in writing to Licensor prior to disclosure as being confidential. Licensor agrees to safeguard all such confidential information ("Licensee Confidential Information") in a manner consistent with the protection accorded to Licensor's own confidential information, and shall restrict access to all Licensee Confidential Information to those employees and consultants of Licensor with a "need to know". Licensor shall have no liability to Licensee for the disclosure of Licensee Confidential Information to third parties except where such disclosure is made willfully and with knowledge that the information disclosed constituted Licensee Confidential Information. Licensor shall return all Licensee Confidential Information to Licensee upon written demand by Licensee, or provide a certificate of destruction with respect thereto.

5. WARRANTY

Unless stated otherwise in Paragraph A, Licensor hereby warrants that it has title to the Product(s), the right to enter into this License Agreement and grant the license hereunder, and that the Product(s), as delivered by Licensor, if properly installed on a Designated CPU in conformance with the written instructions provided to Licensee by Licensor, is capable of operating in conformance with the Product's then-current published specifications. Licensor does not and cannot guarantee the performance or results that may be obtained by use of the Product and Documentation; accordingly, the Product and Documentation is licensed to Licensee "as is." EXCEPT AS SPECIFICALLY PROVIDED IN THIS SECTION, LICENSOR MAKES NO WARRANTIES EITHER EXPRESS OR IMPLIED AS TO ANY MATTER WHATSOEVER, INCLUDING, WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. THE ENTIRE RISK AS TO RESULTS OF USE OF THE PRODUCT IS ASSUMED BY THE USER.

6. INDEMNIFICATION

Licensor agrees to indemnify Licensee and to hold it harmless from all damages awarded against Licensee in the United States or Canada by virtue of Licensee's use of the Product as delivered by Licensor and maintained on a Designated CPU, provided that Licensor is given prompt notice of any such claim and the right to control and direct the investigation, preparation, defense and settlement of each such claim and further provided that Licensee shall fully cooperate with Licensor in connection with the foregoing. Should the Product(s) delivered by Licensor become or, in Licensor's opinion, be likely to become the subject of a claim of infringement of a trade secret, patent or copyright, Licensor may at its option and expense either (a) procure for Licensee the right to continue to use the product as contemplated hereunder, or (b) replace and/or modify the Product to make its use hereunder noninfringing. If neither option is reasonably available to Licensor, then this Agreement may be terminated at the option of either party hereto without further obligation or liability except as provided in Sections 3(C) and 9 hereof, and refund by Licensor to Licensee of the total License Fee actually paid hereunder. Licensor shall have no liability for any claim of trade secret, patent or copyright infringement, based on Licensee's use or combination of the Product with products or data not supplied by Licensor as part of the Product.

7. LIMITATIONS OF LIABILITY AND TIME TO SUE

   a. Modification of product by Licensee. Any modification of the Product by Licensee or any failure by Licensee to implement any Updates to the Product as supplied by Licensor shall void Licensor's maintenance and support obligations under Section 4, Licensor's warranty under Section 5 and Licensor's indemnity under Section 6 above, unless Licensee has obtained prior written authorization from Licensor permitting such modification or failure to implement.

   b. Limitations on Licensor's Liability. Except as provided in section 6 above, Licensor shall not be liable for any direct, indirect, special, consequential or any other damages arising out of Licensee's use of the Product or the marketing, delivery, installation, furnishing, maintaining or supporting of the Product by Licensor. If for any reason any of the foregoing limitations of liability is voided or is not effective, Licensee agrees that (except as provided in Section 6 above) Licensor's liability for damages, if any, shall not exceed the charges paid to Licensor by Licensee for use of the Product under this Agreement. No action, regardless of form, arising out of any of the transactions under this Agreement may be brought by Licensee more than one (1) year after such action accrued.

8. PAYMENT(S); TAXES

   a. Payment. Licensor will invoice Licensee for the amount due on delivery of the Product as specified in Paragraph A. All payments shall be due and payable within thirty (30) days after Licensee's receipt of an invoice from Licensor. Licensee's obligation to pay all accrued charges shall survive the expiration or termination of this Agreement.

   b. Taxes. In addition to all charges specified in this Agreement, Licensee shall pay or reimburse Licensor for all federal, state, local or other taxes not based on Licensor's net income or net worth, including, but not limited to, sales, use, privilege and property taxes, or amounts levied in lieu thereof, based on charges payable under this Agreement or based on the Product, its use or any services performed hereunder, whether such taxes are now or hereafter imposed under the authority of any federal, state, local or other taxing jurisdiction.

9 NO EXPORT

Licensee understands and recognizes that the Product(s) and other materials made available to it hereunder and

4

the direct Product(s) produced through use thereof may be subject to the export administration regulations of the United States Department of Commerce and other United States government regulations relating to the export of technical data and equipment and Product(s) produced therefrom. Licensee is familiar with and agrees to comply with all such regulations, including any future modifications thereof.

10. GENERAL

  a. Notices. All notices, demands or consents required or permitted hereunder shall be in writing and shall be delivered, sent by telegram or telex, or mailed to the respective parties at the addresses set forth in Paragraph A or at such other address as shall have been given to the other party in writing for the purposes of this clause. Such notices and other communications shall be deemed effective upon the earliest to occur of (i) actual delivery, (ii) five (5) days after mailing, addressed and postage prepaid as aforesaid, or (iii) two (2) days after transmission by telex or telegram.

  b. Waiver and Amendment. No waiver, amendment or modification of any provision hereof shall be effective unless in writing and signed by the party against whom such waiver, amendment or modification is sought to be enforced. No failure or delay by either party in exercising any right, power or remedy hereunder shall operate as a waiver of any such right, power or remedy.

  c. Assignment. This Agreement shall be binding upon and inure to the benefit of the successors and assigns of the parties. Licensee may not assign or delegate any of its rights or obligations under this Agreement to any third party without the express written consent of Licensor.

  d. Governing Law. This Agreement shall be governed by the law of the State of California as such law is applied by California courts to contracts between California residents entered into and to be performed within the State of California.

  e. Integration. This Agreement, including any attached Exhibits, constitutes the final, complete and exclusive agreement of the parties concerning the subject matter hereof, and supersedes any other communication related thereto.

  f. Severability. In the event that any provision of this Agreement shall be unenforceable or illegal, such provision shall be severed; and the entire Agreement shall not fail, but the balance of the Agreement shall continue in full force and effect.

ACCEPTED BY LICENSEE: ACCEPTED BY TEKNOWLEDGE, INC.:

Company_____   Teknowledge, Inc.
Address_____   525 University Avenue
_____   Palo Alto, California 94301

Signature_____   Signature_____
Name_____   Name_____
Title_____   Title_____
Date_____   Date_____

TEKNOWLEDGE, INC.
Software License Agreement
Paragraph A

This License Agreement between Teknowledge, Inc. and the Licensee identified below consists of this Paragraph A and the Teknowledge, Inc. General Terms and Conditions Reference Number: ___**51041**___, which is incorporated by this reference.

Date_____

Purchase Order Number_____

The Product(s) to be provided are:

Quantity	Product	Designated CPU Make & Serial #[*]	Maintenance Period	Maintenance Fee	License Fee
_____	S.1 (DEC VAX)	_____ _____			
_____	S.1 (Xerox)	_____			
_____	S.1 (Symbolics)	_____			
_____	M.1	_____			
_____	M.1 Delivery Systems	_____ _____			
_____	M.1a	_____			

[*]Serial number required for S.1 licenses only.

Total Fees: $_____ (exclusive of applicable taxes and shipping charges).

Site(s) location (not required for M.1 Delivery Systems):

(a)_____
    Street Address   City   State   Zip Code

(b)_____
    Street Address   City   State   Zip Code

Accepted by LICENSEE: Accepted by TEKNOWLEDGE, INC.:

Company Name:_____   TEKNOWLEDGE, INC.
Company Address:_____   525 University Avenue, Suite 200
_____   Palo Alto, CA 94301

Signature:_____   Signature_____
Name_____   Name_____
Title_____   Title_____
Date_____   Date_____

Version 6.0 - 9/85

TEKNOWLEDGE, INC.
S.1 General Terms and Conditions

Reference Number: **51041**

1. DEFINITIONS

a. **Agreement.** This Agreement consists of these General Terms and Conditions and Paragraph A attached hereto. These General Terms and Conditions are general terms and conditions for the licensing of Teknowledge's proprietary computer software products. More than one Paragraph A may incorporate these General Terms and Conditions by reference. Each Paragraph A, taken together with these General Terms and Conditions, shall constitute a separate Agreement and shall be considered independent of any other agreements between the parties which incorporate these General Terms and Conditions.

b. **Product.** The term "Product" means one or more of the proprietary computer software programs identified in Paragraph A, all related materials, documentation and information received by Licensee from Licensor and the published specifications for the Product. Paragraph A may identify more than one Product or more than one copy of any Product.

c. **Designated CPU.** The term "Designated CPU" means any central processing unit, including its associated peripheral units, described in Paragraph A. Paragraph A may designate more than one CPU.

2. LICENSE

a. **Grant of License.** Licensor hereby grants to Licensee and Licensee hereby accepts from Licensor a nonexclusive, nontransferable license to use the Product(s) in accordance with this Agreement during the term specified in Section 3. Licensee is aware that this Agreement grants Licensee no title or rights of ownership in the Product and that Licensor considers the Product to be the proprietary information and a trade secret of licensor. It is expressly understood and agreed that the obligations of this section shall survive the expiration or termination of the Agreement or any provision hereof.

b. **Payment of License Fee.** As a condition to Licensor's obligations hereunder, Licensee agrees to pay, within thirty (30) days following receipt of an invoice by Licensor, the one-time License Fee set forth for such Product in Paragraph A.

c. **Restrictions on Use.** Licensee is authorized to use the Product only for Licensee's internal purposes and only on the Designated CPUs at the site(s) specified in Paragraph A. Licensee agrees that it will not use or permit the Product to be used in any manner, whether directly or indirectly, that would enable Licensee's customers or any other person or entity to use the Product on other than a Designated CPU. Licensee acknowledges and agrees that this Subsection 2(C) specifically prohibits, without limitation: (i) incorporation of the Product into any of Licensee's products intended for distribution, or actually distributed, to third parties, (ii) use of the product on or over any computer network system, except one such use for the purpose of initial installation of the Product on the Designated CPU, or thereafter, reinstallations following system failures, (iii)disassembly or copying (except preparation of a single back-up copy) of the Product or (iv) sublicense, commercial time-sharing or rental of the Product. Licensee may not assign or sublicense the license granted hereunder without the prior written consent of Licensor, and any such purported assignment or sublicense shall be void.

d. **Use on Other than Designated CPU.** Notwithstanding the foregoing restrictions on use, but subject to the requirements of notice and consent set forth in this Subsection 2(D), Licensee may use the Product on other than a Designated CPU in the following circumstances: (i) if a Designated CPU cannot be used because of equipment or software malfunctions, Licensee may temporarily use the Product on another CPU; and (ii) if a Designated CPU is replaced by Licensee, Licensee may designate a successor Designated CPU and use the Product on such CPU, provided that use on the prior Designated CPU has ceased. Licensee must give Licensor prior written notice and Licensor must give its prior written consent (which shall not be unreasonably withheld) before such other uses are permitted.

e. **Documentation.** Licensor shall provide Licensee with: (i) the Product in machine readable form and, (ii) Licensor's then-current user documentation for the Product ("Documentation").

f. **Proprietary Markings.** License agrees to maintain all copyright, and/or other markings and/or legends

2

placed upon, contained with or included in the Product, documentation and/or related materials. Such obligation to maintain shall also include the obligation to perpetuate such markings and/or legends on all copies of the Product, documentation and/or related materials prepared by Licensee in accordance with the terms of this Agreement.

3. TERM AND TERMINATION

a. The license granted under this Agreement shall commence upon the delivery of the Product to Licensee and shall continue in perpetuity unless sooner terminated pursuant to Subsection 3(B) below.

b. In the event of any default by the Licensee of any material term covenant, or obligation under this Agreement, this Agreement shall terminate on thirty (30) days prior written notice by Licensor.

c. The following conditions apply upon termination: (i) the Licensee shall discontinue use of the Product and shall deliver to Licensor the Product and related materials furnished by Licensor together with all copies thereof, and shall warrant in writing, within thirty (30) days of termination, that the Product, related materials and all copies thereof have been returned to Licensor; and (ii) the Licensee shall also erase or destroy any of the Product contained in computer memory or data storage apparatus under the control of the Licensee and shall remove the Product from all software systems of Licensee. The Licensee shall warrant such in writing to Licensor within thirty (30) days of termination.

d. If Licensee fails to comply with the provisions of Subsection 3(C) above, Licensor shall have the right to take possession of the Product, wherever the same may be located, upon notice and demand, in accordance with process of law.

4. MAINTENANCE AND SUPPORT

a. Maintenance and Support Services. Subject to the terms, conditions and charges set forth in this section, Licensor will provide Licensee with maintenance and support services for the Product(s) for a period of ninety (90) days from date of invoice as is necessary to cause the Product(s) to perform in accordance with its current published specifications. Licensee acknowledges that access by Licensor to the Designated CPU and Licensee's products being developed with, or containing, the Product(s) shall be essential to performance by Licensor of its obligations under this Section. The maintenance and support services shall be performed by telephone from Licensor's offices in Palo Alto, California or at Licensee's location, as Licensor may elect in its sole discretion.

b. Duration of, and Charges for, Maintenance and Support. There will be no additional charge for maintenance and support during the first ninety (90) days. For each year after the first ninety (90) days Licensor will continue to provide Licensee with maintenance and support services as described in Subsection 4(A) above, provided that Licensee executes a separate Maintenance Agreement and pays Licensor in advance the annual maintenance and support charges then in effect.

c. Limitations on Licensor's Obligations.

Licensee understands and agrees that Licensor may develop and market new or different computer programs which use part or all of the Product(s) and which perform all or part of the functions performed by the Product(s). Nothing contained in this Agreement gives Licensee any rights with respect to such new or different computer programs. Any failure by Licensor to provide ongoing annual maintenance and support on the anniversary date of the agreement shall not constitute grounds for terminating this Agreement but shall be only a basis for terminating the parties' future obligations with respect to maintenance and support.

d. Licensee Confidential Information.

Licensor understands that it may, in the course of performing the maintenance and support services, have access to certain confidential information of Licensee which Licensee has clearly marked as such or otherwise identified in writing to Licensor prior to disclosure as being confidential. Licensor agrees to safeguard all such confidential information ("Licensee Confidential Information") in a manner consistent with the protection accorded to Licensor's own confidential information, and shall restrict access to all Licensee Confidential Information to those employees and consultants of Licensor with a "need to know". Licensor shall have no liability to Licensee for the disclosure of Licensee Confidential Information to third parties except where such disclosure is made willfully and with knowledge that the information disclosed constituted Licensee Confidential Information. Licensor shall return all Licensee Confidential Information to Licensee upon written demand by Licensee, or provide a certificate of destruction with respect thereto.

C — 9

Version 5.0 - 9/85